## Hoofdstuk 2: vrije en open source software

Richard Stallman was in de vroege jaren ’60, al vanaf jonge leeftijd, gefascineerd door computers. Toen hij op zomerkamp was, leende hij programmeerhandleidingen van zijn begeleiders. Er was destijds geen computer te bekennen – ze kostten gemakkelijk meer dan € 100.000 per stuk – maar dat kleine detail ging zijn plezier niet bederven. Tijdens zijn reis schreef hij computerprogramma’s volledig uit op papier.

Het zou nog een paar jaar duren voordat de jonge New Yorker voor het eerst kennismaakte met het echte werk. In 1970, net klaar met de middelbare school, kreeg de toen zeventienjarige Stallman een zomerbaan bij het Wetenschappelijk Centrum van IBM in Manhattan, om een numeriek analyseprogramma te schrijven. Hij rondde het project binnen enkele weken af, wat hem de rest van de zomer de gelegenheid gaf om bij het onderzoekscentrum een teksteditor en een programmeertaalprocessor te ontwerpen, gewoon voor de lol.

Na die zomer schreef Stallman zich in om natuurkunde te studeren aan Harvard. Hij kon doorgaan met programmeren in het vrij nieuwe computercentrum van de universiteit, en begon na verloop van tijd ook rond te kijken naar andere gehoste computers aan verschillende universiteiten en computerfaciliteiten in Cambridge. Hij ontdekte dat een bijzonder krachtige machine gevestigd was in het Kunstmatige Intelligentie Lab (AI Lab) van MIT. Dit onderzoekscentrum van MIT was opgericht door twee pioniers op het gebied van AI - John McCarthy en Marvin Minsky - en werd, zonder extra voorwaarden, gefinancierd door het Amerikaanse ministerie van Defensie.

De student van Harvard besloot dat hij de documentatie van de MIT-computer wilde bestuderen om meer informatie over de machine te verzamelen en te begrijpen hoe deze verschilde van wat ze hadden op Harvard. Maar toen hij het AI Lab bezocht, ontdekte Stallman dat ze helemaal geen dergelijke documentatie hadden.

In plaats daarvan gaven ze hem een baan.

Dit weerspiegelde de tamelijk anarchistische cultuur in het AI Lab. De leiders van het onderzoekscentrum hadden niet veel interesse in ervaring of kwalificaties, maar waardeerden vaardigheid en potentieel. Het was duidelijk dat dit wonderkind van Harvard, die het lab bezocht om hun computerdocumentatie te bestuderen, goed bij hen zou passen.[^30]

### ‘De cultuur van hackers[^31]’

De anarchistische cultuur in het AI Lab was ongeveer een decennium eerder voor het eerst ontstaan.

Het begon toen het Lincoln Lab, een militair onderzoeks- en ontwikkelingscentrum voor geavanceerde technologie verbonden aan het MIT, rond 1960 een kleine revolutie ontketende door de universiteit de TX-0 te schenken, een vroege, volledig getransistoriseerde computer. In tegenstelling tot eerdere computers op de universiteit die altijd een speciale operator nodig hadden, was deze machine voor het eerst toegankelijk voor studenten.

De machine, die een hele kamer in beslag nam en een ton woog, had al snel de fascinatie gewekt van een specifieke groep studenten: de knutselende techneuten uit de modeltreinclub van de universiteit. Ze hadden nooit echt veel interesse gehad in de modeltreinen zelf, maar vonden het vooral leuk om het elektrische systeem van draden, schakelaars en hergebruikte telecomapparatuur te ontwerpen die de snelheid en richting van hun treintjes beheersten. Ze beseften dat er een veel interessanter spel was gearriveerd.

De jonge mannen (het waren aanvankelijk allemaal mannen) waren vastbesloten om de machtige TX-0 te beheersen vanaf het eerste moment dat die op de campus arriveerde. En inderdaad, al snel ontdekten ze hoe ze toegang konden krijgen tot en de broncode konden bewerken van de verschillende programma’s die in de machine waren ingebouwd. Kort daarna wisten ze te achterhalen hoe ze zelf hele nieuwe programma’s konden schrijven.

Het duurde niet lang voordat ze hele nachten rond de TX-0 doorbrachten, op momenten dat ze de machine volledig voor zichzelf hadden. Verenigd door hun gedeelde passie, daagden de jongens elkaar uit om de computer steeds ingewikkeldere taken te laten uitvoeren. Wie pronkte met elegante manieren om code te schrijven, had recht om op te scheppen. Bijzonder slimme oplossingen werden in hun interne jargon ‘hacks’ genoemd; de jongens identificeerden zichzelf dan ook trots als ‘hackers’.

Naarmate de jongens hun programmeervaardigheden verbeterden in een geest van kameraadschap, gingen ze de computer meer en meer zien als een levensstijl. Niets was voor hen belangrijker dan hacken, en niets was leuker. Het benutten van het potentieel van deze krachtige machines gaf hen een geweldig gevoel van zelbeschikking.

Daarmee groeide ook een gevoel van verantwoordelijkheid.

De hackers voelden instinctief aan dat computers een blijvende invloed zouden hebben op de wereld. Na verloop van tijd ontwikkelden ze een filosofisch en ethisch kader rond programmeren en technologie. Dit vormde de basis voor een unieke subcultuur, gericht op technologie en gekenmerkt door experimenten en innovatie. Zelfs toen de groep hackers veranderde – nieuwe studenten kwamen naar MIT, terwijl oudere studenten vertrokken – bleef de hackercultuur behouden.

Als een cruciaal onderdeel van deze cultuur, waren hackers erop gesteld om zaken zelf in handen te nemen: ze wilden alles waarvan ze dachten dat het verbeterd kon worden aanpassen, en herstellen wat kapot was. Om toestemming vragen werd beschouwd als een verspilling van tijd; een goed idee moest onmiddellijk uitgevoerd worden en mogelijke beperkingen moesten worden genegeerd. Bureaucratie was de natuurlijke vijand van de hacker.

Als beperkingen als een uitdaging gezien werden, nou, hackers hielden van uitdagingen, en ze hielden ervan om deze uitdagingen te overwinnen… het liefst met een beetje elegantie en flair. Hackers geloofden dat computers gebruikt konden worden om schoonheid te creëren: code kon esthetische waarde hebben, en hackers bewonderden goed geschreven programma’s en originele oplossingen.

En misschien wel het allerbelangrijkst, ze deelden die.

Hackers waren ervan overtuigd dat vrij toegankelijke code en bestanden iedereen ten goede kwamen, en ze waren trots wanneer mensen de programma’s gebruikten die ze hadden geschreven. Ze geloofden dat ze een ethische plicht hadden om hun code te delen, en om toegang tot informatie voor anderen te vergemakkelijken. Er waren geen wachtwoorden, geen beperkingen, en geen ‘persoonlijke’ documenten.

De hackers zouden uiteindelijk een speciaal besturingssysteem ontwikkelen om dat doel te dienen, het Incompatible Time-Sharing System (ITS) (een woordspeling op het Compatible Time-Sharing System dat eraan voorafging), dat hen in staat stelde om samen te coderen. Als iemand inlogde en ontdekte dat een van zijn kameraden een nieuwe teksteditor of strategisch spel had ontwikkeld, konden ze simpelweg het bestand openen en zelf onmiddellijk bijdragen aan het project. Of, als twee hackers tegelijkertijd de computer gebruikten, konden ze de code gelijktijdig debuggen en verbeteren.

Dit was de vrije en coöperatieve cultuur die Stallman in het AI Lab van MIT ontdekte.

### Anarchisme

De computer van het AI Lab was inderdaad, door iedereen, vrij, en zonder beperkingen, te gebruiken. Niet alleen werknemers, maar ook bezoekers van het lab konden de machine op elk gewenst moment gebruiken, en toegang krijgen tot elk programma of bestand dat zich op de machine bevond. De machine fungeerde als een gedeelde voorziening, voor iedereen beschikbaar.

Het leidde wel tot moeilijkheden. Iedereen die een ITS-computer gebruikte, kon bijvoorbeeld elk bestand verwijderen, zelfs als ze deze niet zelf hadden gemaakt. Op dezelfde manier verstoorde het crashen van de machine elk actief gebruikersproces.

Maar in de praktijk waren dergelijke incidenten zeldzaam. Het vernietigen van iemand anders zijn werk paste niet in de hackercultuur, en hoewel het crashen van de computer hinderlijk was voor andere gebruikers, gaf het hen ook de kans om samen te werken bij het debuggen van de code en het vinden van een oplossing voor wat de crash veroorzaakte.

Dit was heel anders dan de computeromgevingen bij het IBM Wetenschappelijk Centrum en Harvard waaraan Stallman gewend was geraakt. Die machines waren ontworpen met beveiligingsfuncties die vereisten dat sommige mensen meer bevoegdheden hadden dan anderen: bepaalde programma’s waren alleen toegankelijk voor bepaalde gebruikers, zoals systeembeheerders of sommige professoren. De ‘elite’, zij die bevoorrechte accounts hadden, kon eenzijdig beslissen wat anderen wel en niet konden doen op de machines, wat betekende dat reguliere gebruikers vaak om hulp of toestemming moesten vragen.

De nieuwe collega’s van Stallman in het AI Lab walgden van dit soort beleidslijnen. Volgens hen hadden de beheerders in feite politiestaten opgezet in hun respectieve computeromgevingen, waarbij ze zichzelf de autoriteit toe-eigenden om andere gebruikers te besturen en controleren.

Nu hij hun vrije alternatief had ervaren, was Stallman het volledig met hen eens. Terwijl het AI Lab bewees dat hun vorm van anarchie een productieve werkomgeving kon bevorderen, raakte hij ervan overtuigd dat de beperkende en gecontroleerde systemen feitelijk een digitale vorm van fascisme vertegenwoordigden.[^32]

‘De gebruikers van ons systeem waren vrije mensen, aan wie gevraagd werd om zich verantwoordelijk te gedragen. In plaats van een elite van macht, hadden we een elite van kennis, bestaande uit iedereen die gemotiveerd was om te leren’, schreef Stallman later. ‘Omdat niemand anderen kon domineren op onze machine, functioneerde het lab als een anarchie. Het zichtbare succes hiervan bekeerde mij tot het anarchisme. Voor de meeste mensen betekent ‘anarchie’ ‘verspillende, destructieve wanorde’, maar voor een anarchist als ik betekent het vrijwillige organisatie naar behoefte, met de nadruk op doelen, in tegenstelling tot regels en vereisten tot uniformiteit omwille van de uniformiteit.’[^33]

Hoewel Stallman geen anarchist was in de meest complete betekenis van het woord – hij geloofde namelijk nog steeds dat de staat vele belangrijke functies uitvoerde, waaronder het financieren van het AI Lab – dacht hij dat het anarchistische model ook in andere computeromgevingen kon werken. En inderdaad, rond deze tijd begon de hackercultuur zich ook te verspreiden buiten MIT, met name naar Stanford University, die een eigen AI Lab kreeg. Tegen het begin van de jaren ’70 had de hackergemeenschap een nieuwe basis gevestigd in de San Francisco Bay Area.

En Stallman geloofde dat de hackercultuur ook buiten het academische domein levensvatbaar zou zijn. Met het AI Lab als een geslaagd voorbeeld, zou het vrije en samenwerkende ethos wellicht een model kunnen worden voor de opkomende computerrevolutie.

### Problemen in het paradijs

Later werd echter duidelijk dat de verspreiding van de hackercultuur niet zo eenvoudig was.

Bijna tien jaar later was Stallman nog steeds werkzaam in het lab. Hij merkte dat de hackercultuur daadwerkelijk verdreven begon te worden uit haar oorspronkelijke thuisbasis. Mensen in en rond het AI Lab gingen steeds meer wachtwoorden en, erger nog, auteursrechtelijke licenties omarmen. Tegelijkertijd wilden de beheerders van MIT dat computergebruikers formulieren invulden voordat ze de machines konden bedienen, een praktijk die Stallman actief probeerde tegen te houden.[^34]

En toch, vergeleken met wat er ging komen, waren dit kleine problemen.

In 1979 wilden Richard Greenblatt, een van de meest gerespecteerde hackers van het lab, en Russell Noftsker, een voormalig labbeheerder, een van de meest prominente projecten van het AI Lab op de markt brengen. Hun plan was om een start-up op te richten om speciale computers te verkopen die ontworpen waren voor LISP, de programmeertaal voor AI die in het onderzoeksinstituut in ontwikkeling was.

Het werd echter al snel duidelijk dat Greenblatt en Noftsker zeer verschillende ideeën hadden over de start-up. Greenblatt wilde dicht bij de geest van het AI Lab en haar anarchistische cultuur blijven. Dit betekende dat hij uit de buurt wilde blijven van investeerders en zo dicht mogelijk wilde aanleunen bij de gekende hackercultuur. Noftsker vond de benadering van Greenblatt echter onrealistisch. Hij zag een meer traditioneel bedrijf voor zich, dat zijn producten zou beschermen met softwarelicenties en auteursrechten.

Greenblatt en Noftsker slaagden er niet in om een compromis te bereiken en besloten uit elkaar te gaan. Ze startten ieder hun eigen onderneming: Greenblatts LISP Machine Incorporated (LMI) en Noftskers Symbolics werden rivalen.

Aanvankelijk deelden zowel LMI als Symbolics de code die ze bij het AI Lab produceerden, en daardoor ook met elkaar. Echter, begin 1982, verbrak Symbolics deze driehoeksrelatie. Noftsker besloot dat het AI Lab wel nog de door Symbolics aangepaste versie van de LISP-software kon gebruiken, maar LMI mocht dit niet meer. Het was een ultimatum. De beslissing van Noftsker betekende dat elke hacker in het lab een kant moest kiezen.

Hoewel Greenblatt, die veel werk had verricht om het LISP-project te realiseren, de kennis en de capaciteiten had, beschikte hij over te weinig middelen. Ondertussen had het bedrijfsplan van Symbolics Noftsker in staat gesteld om fondsen te werven van investeerders. Hij gebruikte het geld om enkele van de beste hackers van het AI Lab in te huren. Om er zeker van te zijn dat de nieuw aangeworven hackers exclusief voor zijn start-up zouden werken, verbood hij alle medewerkers van Symbolics om bij te dragen aan het AI Lab.

In één klap waren veel van de beste programmeurs van het computerlab verdwenen, en ze namen hun werk met hen mee.

Het AI Lab was effectief uitgekocht. MIT’s toevluchtsoord voor vrije samenwerking was frontaal in botsing gekomen met meedogenloze zakelijke belangen. Omdat de kleinschalige utopie van de hackers geroofd was van haar meest waardevolle middelen, bleef slechts een uitgehold overblijfsel van het onderzoekscentrum achter. Het lab had tijdens een kortstondig gouden tijdperk gediend als een toonbeeld van effectief anarchisme, maar was dat na de beslissing van Noftsker niet langer.

Voor Stallman betekende dit het einde van het lab.

De ontgoochelde hacker vatte het kort daarna samen in een brief:

‘De personen die nog in het lab waren, waren de professoren, studenten en non-hackeronderzoekers, die niet wisten hoe ze het systeem of de hardware moesten onderhouden, of dit zelfs niet wilden weten. Machines begonnen te breken en werden nooit gefixt; soms werden ze gewoon weggegooid. Noodzakelijke veranderingen in software konden niet worden doorgevoerd. De niet-hackers reageerden hierop door over te schakelen naar commerciële systemen, wat fascisme en licentieovereenkomsten met zich meebracht. Ik slenterde door het lab, door de kamers die ‘s nachts leeg waar ze vroeger vol waren en dacht: ’Oh mijn arme AI-lab! Je gaat dood en ik kan je niet redden.’ Iedereen verwachtte dat als er meer hackers werden opgeleid, Symbolics ze zou wegkapen, dus het leek niet eens de moeite waard om het te proberen… de hele cultuur werd uitgewist…’[^35]

Stallman had ooit gedroomd van een toekomst geïnspireerd door de vrije en collaboratieve hackercultuur, maar hij geloofde nu dat hij in plaats daarvan de laatste ademtochten ervan aan het aanschouwen was.

‘Ik ben de laatste overlevende van een dode cultuur’, klaagde Stallman met een gevoel van drama. ‘En ik hoor echt niet meer thuis in de wereld. En op sommige manieren voel ik dat ik eigenlijk dood zou moeten zijn.’[^36]

### Vrije software

Toch was Stallman nog niet helemaal bereid om op te geven.

Stallman wees voornamelijk Noftsker aan als schuldige voor de ondergang van het AI Lab. De hacker zette zich vervolgens in om alle software-upgrades van Symbolics opnieuw te implementeren. Hij hield hun documentatie van nieuwe functies bij en schreef vervolgens code die dezelfde functies bood. In feite deed hij in zijn eentje het werk van zes ontwikkelaars van de startup. Hij deelde zijn code met LMI, waardoor het bedrijf van Greenblatt een kans had om tegen Symbolics te concurreren. Hij hield dit lang genoeg vol zodat Greenblatt nieuwe programmeurs kon aannemen en zijn bedrijf weer op de rails kon krijgen.[^37]

Vervolgens besloot Stallman dat het tijd was voor een nieuwe start. Hij had zichzelf ervan overtuigd dat de hackercultuur de wereld nog steeds kon veranderen, maar concludeerde dat er een nieuw plan nodig was: ‘een ambitieus project dat de fundamenten aanvalt van de manier waarop de commerciële, vijandige manier van leven wordt voortgezet.’[^38]

Specifiek wilde Stallman de algemene trend naar *propriëtaire software* omkeren, software die door licenties en auteursrechten beperkt werd, en die in de jaren ’80 steeds gebruikelijker werd. In lijn met de hackerethiek geloofde Stallman dat een computerprogramma maximaal nut biedt als mensen het kunnen verbeteren. En aangezien computers het quasi kosteloos maakten om informatie te kopiëren, stelde hij dat het verhinderen van het delen van software ‘de gehele mensheid saboteert’, zo betoogde de hacker.[^39]

Erger nog, propriëtaire software kan doorgaans niet geïnspecteerd worden. Als mensen geen toegang hebben tot de in mensentaal leesbare broncode van de software die ze op hun computers draaien, kunnen ze niet zeker weten wat hun eigen machine eigenlijk doet. Een programma kan kwaadaardig zijn en bijvoorbeeld zijn gebruiker beperken, censureren, bespioneren of op een andere manier misbruiken.[^40]

Stallman was van mening dat het gebruik van propriëtaire software in feite betekende dat je de controle overliet aan degene die het had ontwikkeld.

‘Als gebruikers het programma niet beheersen, beheerst het programma de gebruikers’, redeneerde hij. ‘Met eigendomsrechtelijk beschermde software is er altijd een entiteit – de ‘eigenaar’ van het programma – die controle heeft over het programma en daarmee macht uitoefent over de gebruikers.’[^42]

In plaats daarvan wilde Stallman dat computers instrumenten van zelfbeschikking en vrijheid waren. Hij geloofde dat gebruikers te allen tijde de controle over hun eigen apparaten moesten hebben.

Om dit te verwezenlijken, ontwikkelde hij een filosofie die vereiste dat elk computerprogramma vier essentiële vrijheden moest bieden:[^43]

* De vrijheid om het programma te gebruiken zoals je wenst, voor welk doel dan ook (vrijheid 0)[^44].
* De vrijheid om te bekijken hoe het programma werkt en het zo aan te passen dat het je computerwerk naar wens uitvoert (vrijheid 1). Toegang tot de broncode is een noodzakelijke voorwaarde om dit te realiseren.
* De vrijheid om kopieën te herverdelen zodat je anderen kunt helpen (vrijheid 2).
* De vrijheid om kopieën van je aangepaste versies met anderen te delen (vrijheid 3). Door dit te doen kun je de hele gemeenschap de kans geven om te profiteren van je veranderingen. Toegang tot de broncode is een noodzakelijke voorwaarde.

Of samengevat, gebruikers hadden over het algemeen ‘de vrijheid om software te draaien, kopiëren, verspreiden, bestuderen, veranderen en te verbeteren.’[^45] Dit vereist dat de menselijk-leesbare broncode van een programma gepubliceerd is, en dat deze niet onderhevig is aan beperkende auteursrechtenlicenties. Stallman zou software, die deze vier essentiële vrijheden bood, classificeren als vrije software (vrij in de betekenis van ‘vrijheid’, benadrukte de hacker graag, en niet zoals in ‘gratis bier’.).

### GNU

Om echt de belofte van vrije software waar te maken, begreep Stallman dat elk programma op een computer de vereiste vrijheden moest bieden. Dit omvatte – in de eerste plaats – het besturingssysteem. Een tekstverwerker die zich houdt aan de vier vrijheden kan zijn gebruiker niet stiekem bespioneren, maar als het besturingssysteem waarop de tekstverwerker draait ook niet-vrije software is, kan niet worden uitgesloten dat het besturingssysteem bespioneert.

Daarom kondigde Stallman in 1983 zijn ongelooflijk ambitieus project aan om een alternatief te bieden voor het populaire Unix besturingssysteem. Waar Unix propriëtaire software was, bestond Stallmans besturingssysteem volledig uit vrije software. Passend noemde hij het project GNU: GNU is Niet Unix! (inderdaad, een recursief acroniem).[^46]

GNU belichaamde de hackerethiek en wees propriëtaire software volledig af.

‘Ik heb veel andere programmeurs gevonden die enthousiast zijn over GNU en willen helpen’, schreef Stallman in het GNU Manifesto dat hij uitgaf na de aankondiging, waarin hij het doel en de staat van het project beschreef. ‘Veel programmeurs zijn ongelukkig over de commercialisering van systeemsoftware. Het stelt hen wellicht in staat om meer geld te verdienen, maar het zorgt ervoor dat ze zich in conflict voelen met andere programmeurs in plaats van als kameraden. […] GNU dient als voorbeeld om te inspireren en als een vaandel om anderen rond te verzamelen om ons te vergezellen in het delen.’[^47]

Inderdaad, GNU was meer dan alleen een stukje software, maar vertegenwoordigde het onstaan van een nieuwe sociale beweging: de vrije softwarebeweging.

Om de beweging te ondersteunen, richtte Stallman in 1985 ook de non-profitorganisatie Free Software Foundation op. De stichting zou pleiten voor vrije software en geld inzamelen om vrije softwareprojecten te financieren. Daarnaast leidde de Free Software Foundation de introductie van speciale vrije softwarelicenties onder de nieuwe paraplu van ‘copyleft’, ontworpen om vrije software te stimuleren.

Dit omvatte, het meest opmerkelijk, de GNU General Public License: een licentie die het recht verleent om broncode te verspreiden en te wijzigen zolang dit wordt gedaan onder even vrije voorwaarden. Met andere woorden, andere ontwikkelaars van vrije software konden vrije software die onder deze licentie werd uitgebracht op welke manier dan ook in hun eigen project integreren, maar ontwikkelaars van propriëtaire software konden dit niet.

Met het GNU-project in volle gang en de nieuwe licenties geïmplementeerd, stond vrije software op het punt om een kracht te worden waar je maar best rekening mee hield.

### De kathedraal en de bazaar

Traditioneel werden vrije softwareprojecten uitgevoerd door kleine groepjes ontwikkelaars vanuit speciale technologiehubs, zoals het AI Lab van MIT. Maar toen hij aan GNU begon te werken, nodigde Stallman andere ontwikkelaars uit om ook aan zijn project mee te werken. Door gebruik te maken van het ontluikende internet, konden hackers zelfs vanuit de hele wereld code bijdragen.

Hoewel Stallman aan bijdragers voor hun werk normaal geen financiële compensatie beloofde, waren veel ontwikkelaars desondanks bereid om te helpen van GNU een realiteit te maken. Misschien hoopten sommigen van hen respect of status te verdienen van hun programmeercollega’s door bij te dragen, zoals altijd een factor was binnen de hacker-gemeenschap. Anderen droegen misschien bij omdat ze GNU zelf wilden gebruiken. Nog anderen vonden misschien de uitdaging op zichzelf interessant genoeg om er deel van uit te maken. En, misschien wilden sommigen gewoon de wereld een betere plaats maken, en zagen dit project als een middel om dat doel te bereiken.

Wat hun redenen ook waren, ze droegen bij. Wat nog meer is, hun bijdragen waren waardevol. Deze vrijwillige programmeurs leverden, enigszins opmerkelijk, hoogwaardige code, waardoor Stallman in staat was om vele afzonderlijke delen van het GNU-besturingssysteem een paar jaar later te voltooien — een indrukwekkende prestatie.

Rond datzelfde moment maakte de Finse software-ingenieur Linus Torvalds graag gebruik van de vrijheid geboden door de GNU General Public License. Hij nam een groot deel van Stallmans GNU-code, maar voegde zijn eigen kernel toe (een programma dat zich in het hart van een besturingssysteem van een computer bevindt), en in 1992 bracht Torvalds Linux uit.[^48] Het was het eerste werkende besturingssysteem dat volledig uit vrije software bestond.

Maar Torvalds’ voornaamste vernieuwing was mogelijk niet het Linux-kernel zelf. Het was de manier waarop hij het maakte. De jaren ervoor ontwikkelde de software-ingenieur een proces dat expliciet is ontworpen voor samenwerking via het internet.

Zoals uitvoerig besproken door Linux-bijdrager Eric S. Raymond in zijn essay ‘The Cathedral & The Bazaar’ uit 1997 en het (later) gelijknamige boek, was de grootste aanpassing die Torvalds maakte de benadering van het project omtrent beveiliging.

Tot die tijd beschouwden vrije softwareontwikkelaars bugs en andere kwetsbaarheden als grote risico’s die moesten worden aangepakt door toegewijde experts die hun software zorgvuldig beoordeelden. Dit omvatte ook de code die ze ontvingen van externe bijdragers. Ze zouden de code pas vrijgeven als ze er zeker van waren dat het veilig was om deze te gebruiken. Raymond bestempelde deze top-down benadering als het ‘kathedraal’model.

In plaats daarvan gebruikte Torvalds wat Raymond het ‘bazaar’model noemde. Dit model gebruikte een flexibelere methode om bijdragen te integreren, waardoor ontwikkelaars meer direct hun wijzigingen konden uploaden naar verschillende versies van de software. Andere bijdragers konden vervolgens deze software downloaden, testen en mogelijk de wijzigingen in hun eigen versies overnemen.

Zo’n flexibel systeem zou kunnen leiden tot versies van de software met meer fouten dan de software van hun tegenhangers in het kathedraal-model. Maar omdat het ontwikkelproces openlijk plaatsvindt, zijn andere bijdragers meestal sneller in staat om deze fouten te ontdekken en ze te corrigeren. Indien nodig, wordt de oplossing direct opgenomen in een nieuwe versie; onder het bazaarmodel vinden software-updates sneller en vaker plaats.

‘Met een groot genoeg aantal bèta-testers en mede-ontwikkelaars, wordt bijna elk probleem snel vastgesteld en is de oplossing voor iemand duidelijk’, schreef Raymond in zijn essay, waarbij hij een van de belangrijkste lessen die hij over de jaren had geleerd, samenvatte. ‘Of, minder formeel, “Met genoeg ogen, zijn alle bugs oppervlakkig.”’[^49]

Hij gaf de uitspraak de naam de ‘Wet van Linus’.

Opmerkelijk genoeg, geloofde Raymond dat dit ontwikkelingsmodel voordelen kon bieden zelfs aan bedrijven en mensen die Stallmans bezorgdheid over eigendomssoftware niet deelden, maar gewoon kwaliteitscode wilden tegen lage kosten. Hij vermoedde echter dat velen van hen (vooral bedrijven) terughoudend waren om gratis software te gebruiken, juist omdat ze afgeschrikt werden door de ideologische verhalen eromheen. Om minder nadruk te leggen op Stallmans originele motivaties en meer op de pragmatische voordelen, leidde Raymond daarom in de late jaren ’90 de poging om vrije software te rebranden als ‘open source software’.

Stallman was zelf echter niet akkoord met de herbenoeming. Voor hem was vrijheid het voornaamste, en de term ‘open source’ deed afbreuk aan deze boodschap.

Vandaag de dag verwijzen de termen ‘vrije software’ en ‘open source software’ in vrijwel alle gevallen naar hetzelfde concept, maar het verschil in terminologie blijft de filosofische kloof vertegenwoordigen. De term ‘vrije en open source software’ (FOSS) wordt gebruikt om beide zijden van het schisma expliciet te omvatten.[^50]

### Gemeenschappelijk begrip

Het bazaarmodel kan hoogwaardige code opleveren. Maar die kwaliteit is geen vanzelfsprekendheid. Volgens de Wet van Linus, vereist hoogwaardige code voldoende ‘ogen’, oftewel bijdragers.

FOSS-projecten hebben meestal niet de middelen om financiële beloningen te geven aan potentiële bijdragers. Daarnaast worden bestaande machtsverhoudingen vaak genegeerd in het kader van vrije en open-source-ontwikkeling. Zoals Raymond ook al uitsprak in zijn essay, was dwang natuurlijk volledig buiten de orde in ‘het anarchistische paradijs dat we het internet noemen.’ [^51] Het aantrekken van bijdragers is daarom een cruciale vaardigheid gebleken voor ontwikkelaars van vrije en open software.

Geïnspireerd door de negentiende-eeuwse Russische anarchist Pyotr Alexeyevich Kropotkin, legde Raymond uit dat projectleiders moesten leren om effectieve belangengemeenschappen te werven en te motiveren op basis van een gemeenschappelijk begrip. Om ontwikkelaars te overtuigen een bijdrage te leveren, moet de leiding van een project bedenken hoe bij te dragen aan hoe het project hen ten goede zou komen. De prikkels moeten worden afgestemd op een gedeeld doel, stelde Raymond voor, een ‘ernstige inspanning van vele samenkomende wilskrachten.’[^52]

Dit betekent in de praktijk dat niemand echt de leiding heeft over FOSS-projecten in de bazaarstijl. Een projectleider kan het project niet in een richting sturen die niet gedragen wordt door de rest, zonder de ontwikkelaars te verliezen die hij zo hard nodig heeft. In het bazaarmodel wordt software beheerd door zijn schare van bijdragers, elk met hun eigen persoonlijke reden om betrokken te zijn.

Wanneer deze prikkels wel in lijn liggen en er een groep bijdragers bereid is aan een gemeenschappelijk doel te werken, kunnen de resultaten geweldig zijn. Hoewel niemand ooit echt de leiding heeft, zijn deze grootschalige samenwerkingsverbanden tussen vreemden met sterk uiteenlopende niveaus van kennis en vaardigheden, erin geslaagd om zeer complexe programma’s te produceren, waarvan de Linux-kernel slechts één voorbeeld is van vele.

Op deze manier lijkt de ontwikkeling van vrije en open source software sterk op die andere vorm van grootschalige, leiderloze samenwerking: vrije markten. Net zoals vrije markten bestaan FOSS-projecten alleen uit vrijwillige interacties, benutten ze de kennis die verspreid is onder de deelnemers en wat misschien wel het meest interessant is, kunnen ze beter presteren dan top-down organisatievormen.

Net zoals vrije markten, kunnen vrije en open softwareprojecten een spontane orde vormen.

‘In veel opzichten gedraagt de Linux-wereld zich als een vrije markt of een ecologie, een verzameling van individuen die hun eigenbelang nastreven. Tijdens dit proces ontstaat een zichzelf corrigerende spontane orde die veel complexer en efficiënter is dan wat elke vorm van centrale planning ooit zou kunnen bereiken.’[^53]