5

GELD

In dit hoofdstuk wordt gekeken hoe een open source-softwareomgeving aan fondsen kan komen. Dit is niet alleen bestemd voor ontwikkelaars die worden betaald om aan open source-softwareprojecten te werken, maar ook voor hun managers die inzicht moeten hebben in de sociale dynamiek van de ontwikkelomgeving. In dit hoofdstuk wordt ervan uitgegaan dat de aangesproken persoon ('u') een betaalde ontwikkelaar is, of iemand die leiding geeft aan deze ontwikkelaars. Voor beiden is het advies vaak hetzelfde en wanneer dit niet zo is, wordt in de context duidelijk gemaakt voor wie het bedoeld is.

Commerciële financiering van de ontwikkeling van open source-software is geen nieuw verschijnsel. Veel van deze ontwikkeling is altijd op informele wijze bekostigd. Wanneer een systeembeheerder een netwerkanalyse-instrument schrijft om hem te helpen zijn werk te doen, dit vervolgens online zet en bugfixes en functiebijdragen krijgt van andere systeembeheerders, is er in feite een informeel consortium gevormd. De financiële middelen van het consortium komen uit het salaris van de systeembeheerders. Kantoorruimte en netwerkbandbreedte worden, zonder dat men dit beseft, gedoneerd door de organisaties waarvoor zij werken. Die organisaties hebben natuurlijk baat bij de investering, ook al zijn ze zich daar in het begin niet van bewust.

Het verschil is dat tegenwoordig veel van deze inspanningen worden geformaliseerd. Bedrijven zijn zich bewust geworden van de voordelen van open source-software en zijn zelf directer betrokken geraakt bij de ontwikkeling ervan. Ook ontwikkelaars zijn gaan verwachten dat zeer belangrijke projecten ten minste donaties aantrekken en mogelijk zelfs lange-termijnsponsors. Hoewel de aanwezigheid van geld de basisdynamiek van open source-softwareontwikkeling niet heeft gewijzigd, heeft het wel in hoge mate de schaal veranderd waarop dingen gebeuren, zowel wat betreft het aantal ontwikkelaars als de tijd per ontwikkelaar. Het heeft tevens invloed gehad op de wijze waarop projecten worden georganiseerd en op de interactie van de daarbij betrokken partijen. Het gaat er niet alleen om hoe het geld wordt uitgegeven of hoe het rendement op de investering wordt gemeten. Het gaat tevens over management en processen: hoe kunnen de hiërarchische leidinggevende structuren van bedrijven en de semigedecentraliseerde commissies van vrijwilligers van open

source-softwareprojecten op productieve wijze met elkaar samenwerken? Zullen ze het eens kunnen worden over de betekenis van 'op productieve wijze'?

Financiële steun wordt doorgaans toegejuicht door open source-ontwikkelgemeenschappen. Het kan de kwetsbaarheid van een project voor de 'macht van de chaos' verminderen waardoor zoveel projecten worden weggevaagd voordat ze echt van de grond zijn gekomen. Daardoor kan het mensen bereidwilliger maken om de software een kans te geven: ze hebben het gevoel dat ze hun tijd investeren in iets wat er over zes maanden nog steeds zal zijn. Geloofwaardigheid immers, is tot op zekere hoogte aanstekelijk. Als bijvoorbeeld IBM een open source-project steunt, gaan mensen er gevoeglijk vanuit dat het project niet zal mislukken. De daaruit voortvloeiende bereidheid om zich voor het project in te zetten, kan ertoe bijdragen dat de voorspelling zichzelf verwezenlijkt.

Financiering gaat echter ook gepaard met het concept controle. Als niet voorzichtig met het fenomeen geld wordt omgegaan, kan het een project verdelen in twee groepen, groepsontwikkelaars en ontwikkelaars buiten de groep. Als de onbetaalde vrijwilligers het gevoel krijgen dat ontwerpbesluiten of functietoevoegingen eenvoudigweg beschikbaar zijn voor de hoogste bieder, stappen ze over op een project dat meer op een meritocratie lijkt en minder op onbetaald werk ten gunste van iemand anders. Het kan zijn dat ze nooit openlijk klagen op de mailinglijsten. In plaats daarvan houden de vrijwilligers geleidelijk op met hun pogingen om serieus te worden genomen en komt er daardoor steeds minder geluid vanuit externe bronnen. De bedrijvigheid van kleinschalige activiteiten gaat door, in de vorm van bugrapporten en af en toe wat kleine fixes. Maar er is dan geen sprake van grote codebijdragen of externe deelname aan ontwerpbesprekingen. Mensen voelen wat van hen wordt verwacht en voldoen (of niet) aan deze verwachtingen.

Hoewel geld voorzichtig moet worden gebruikt, betekent dat niet dat het geen invloed kan kopen. Dat kan het namelijk beslist wel. De essentie is dat geld niet rechtstreeks invloed kan kopen. Bij een simpele handelstransactie ruil je geld voor datgene wat je wilt hebben. Als je een extra functie nodig hebt, onderteken je een contract, betaal je ervoor en wordt het gedaan. Bij een open source-project ligt dat niet zo eenvoudig. U kunt een overeenkomst sluiten met een aantal ontwikkelaars, maar ze zouden zichzelf – en u – voor de gek houden als ze zouden garanderen dat het werk waarvoor u hebt betaald door de ontwikkelaarsgemeenschap zou worden aanvaard alleen maar omdat u ervoor hebt betaald. Het werk kan alleen worden aanvaard om de intrinsieke waarde ervan en om hoe het past in de visie die de open source-gemeenschap van de software heeft. Mogelijk hebt u wat inbreng in die visie, maar u bent niet de enige.

Geld kan dus geen invloed kopen, maar het kan wel zaken kopen die tot invloed *leiden*. Het duidelijkste voorbeeld zijn programmeurs. Als er goede programmeurs worden ingehuurd, die lang genoeg blijven om ervaring met de software op te doen en geloofwaardigheid bij de gemeenschap te krijgen, kunnen deze via dezelfde middelen het project beïnvloeden als alle andere leden. Ze hebben een stem, of als ze met velen zijn, hebben ze een stemcoalitie. Als ze in het project gerespecteerd worden, hebben ze meer invloed dan alleen via hun stemmen. Betaalde ontwikkelaars hoeven hun motieven ook niet te verbergen. Iedereen die wil dat er een verandering in de software wordt aangebracht, wil dat immers om een bepaalde reden. De beweegredenen van uw bedrijf zijn niet minder legitiem dan die van ieder ander. Het is alleen zo dat het belang dat aan de doelen van uw bedrijf wordt gehecht, wordt bepaald door de status van de vertegenwoordigers daarvan in het project en niet door de omvang, het budget of het ondernemingsplan van het bedrijf.

5.1 SOORTEN BETROKKENHEID

Er zijn veel verschillende redenen waarom open source-projecten worden gefinancierd. De onderdelen op deze lijst sluiten elkaar niet uit. Vaak is de financiële steun van een project het gevolg van verschillende van deze redenen, of van alle.

De lasten delen

Afzonderlijke organisaties met min of meer dezelfde softwarebehoeften doen het werk vaak dubbel, hetzij doordat ze intern dezelfde (dus overbodige) code obtwikkelen, hetzij door soortgelijke gesloten producten in te kopen van commerciële leveranciers. Wanneer ze beseffen wat er aan de hand is, kunnen de organisaties hun krachten bundelen en een open source-project creëren (of zich daarbij aansluiten) dat past bij hun behoeften. De voordelen zijn duidelijk: de ontwikkelkosten worden gedeeld en iedereen heeft er baat bij. Hoewel dit scenario het meest logisch lijkt voor non-profitorganisaties, kan het ook strategisch zinvol zijn voor concurrenten met een winstoogmerk.

Voorbeelden: http://www.openadapter.org/, http://www.koha.org/

Uitbreiding van diensten

Wanneer een bedrijf diensten verkoopt die afhankelijk zijn van, of aantrekkelijker worden door, bepaalde open source-programma's, is het uiteraard in het belang van dat bedrijf om ervoor te zorgen dat die programma's actief worden onderhouden.

Voorbeeld: CollabNets ondersteuning van http://subversion.tigris.org/ (disclaimer: dat is mijn dagelijkse werk, maar het is ook een fraai voorbeeld van dit model).

Ondersteuning van de verkoop van hardware

De waarde van computers en computeronderdelen houdt direct verband met de hoeveelheid software die ervoor beschikbaar is. Hardwareleveranciers – niet alleen leveranciers van hele machines, maar ook makers van randapparatuur en microchips – hebben ontdekt dat het voor klanten van belang is dat zij beschikken over open source-software van goede kwaliteit die gebruikt kan worden op hun hardware.

Ondermijning van een concurrent

Soms ondersteunen bedrijven een bepaald open source-project als middel om het product van een concurrent te ondermijnen, dat zelf al dan niet open source is.

Het afsnoepen van marktaandeel van een concurrent is doorgaans niet de enige reden om betrokken te raken bij een open source-project, maar het kan een factor zijn.

Voorbeeld: http://www.openoffice.org/ (nee, dit is niet de enige reden dat OpenOffice bestaat, maar de software is, althans deels, een reactie op Microsoft Office).

Marketing
Het kan een goede marktstrategie zijn om uw bedrijf te koppelen aan een populaire open source-toepassing.

Dubbele licentie
Dubbele licentie is de praktijk van het aanbieden van software onder een traditionele octrooilicentie voor klanten die deze willen doorverkopen als onderdeel van een gepatenteerde toepassing van henzelf en tegelijkertijd onder een vrije licentie voor degenen die deze willen gebruiken onder open source-voorwaarden (zie het gedeelte 'Modellen voor dubbele licenties' in Hoofdstuk 9, *Licenties, auteursrechten en octrooien*). Als de open source-ontwikkelaarsgemeenschap actief is, kan de software profiteren van debugging en ontwikkeling op grote schaal, terwijl het bedrijf nog steeds inkomsten ontvangt uit de opbrengst van rechten om enkele fulltime programmeurs te ondersteunen.

Twee bekende voorbeelden zijn MySQL, makers van de gelijknamige databasesoftware, en Sleepycat, die distributies en ondersteuning voor de Berkeley Database biedt. Het is geen toeval dat dit beide databasebedrijven zijn. Databasesoftware wordt vaker in toepassingen geïntegreerd dan dat deze rechtstreeks aan de gebruiker wordt verkocht en is dus erg geschikt voor het model van dubbele licentie.

Donaties
Een op grote schaal gebruikt project ontvangt soms grote bedragen van zowel afzonderlijke personen als organisaties, gewoon via een online donatieknop of, soms, door middel van merchandising van koffiemokken, T-shirts, muismatjes enz. met het merk erop. Enige voorzichtigheid is geboden. Als uw project donaties accepteert, plan dan de besteding van het geld *voordat* het binnenkomt en vermeld de plannen op de website van het project. Discussies over de toewijzing van geld verlopen doorgaans veel soepeler wanneer ze worden gehouden voordat er daadwerkelijk geld is om uit te geven. Bovendien is het beter om in het geval van grote onenigheid daar achter te komen zolang alles nog theoretisch is.

Het bedrijfsmodel van de financier is niet de enige factor in de relatie tot een open source-gemeenschap. Ook de historische relatie tussen beide is van belang: is het bedrijf met het project begonnen, of sluit het zich aan bij een bestaand ontwikkelingsproject? In beide gevallen moet de financier geloofwaardigheid verdienen, maar het zal geen verbazing wekken dat er in het laatste geval iets meer moet worden verdiend. De organisatie moet duidelijke doelen hebben met betrekking tot het project. Probeert het bedrijf zijn leidende positie te behouden, of probeert het eenvoudigweg een stem in de gemeenschap te zijn om de richting van het project te leiden maar niet noodzakelijkerwijs te besturen? Of wil het slechts over een paar committers beschikken die bugs van klanten kunnen repareren en de wijzigingen zonder gedoe in de openbare distributie kunnen krijgen?

Houd deze vragen in gedachten wanneer u de richtlijnen hieronder leest. Deze gelden voor de toepassing van iedere soort betrokkenheid van de organisatie bij een open source-softwareproject, maar elk project is een menselijke omgeving en daarom zijn geen twee projecten precies gelijk. In zekere zin moet u altijd kijken hoe het loopt, maar als u deze principes volgt, is het waarschijnlijker dat dingen gaan lopen zoals u wilt.

5.2 AANSTELLING VOOR DE LANGE TERMIJN

Als u programmeurs bij een open source-project aanstuurt, zorg dan dat u hen daar lang genoeg houdt, zodat ze zowel technische als politieke ervaring opdoen: ten minste twee jaar. Geen enkel project, of dit nu een open source- of closed source-project is, is er natuurlijk bij gebaat als er te vaak van programmeurs wordt gewisseld. De noodzaak om telkens nieuwkomers aan te trekken die alles weer van de grond af moeten leren, werkt afschrikwekkend, ongeacht de omgeving. Maar de nadelen zijn nog groter voor een open source-project, omdat ontwikkelaars die vertrekken niet alleen hun kennis van de code met zich meenemen, maar tevens hun status in de gemeenschap en de informele contacten die ze daar zijn aangegaan.

De geloofwaardigheid die een ontwikkelaar heeft verworven, kan niet worden overgedragen. Om het meest logische voorbeeld te geven: een nieuw binnengekomen ontwikkelaar kan de commit access van een vertrekkende ontwikkelaar niet overnemen (zie het gedeelte 'Met geld kun je geen liefde kopen' verderop in dit hoofdstuk). Dus als de nieuwe ontwikkelaar niet reeds commit access heeft, moet hij patches indienen totdat hij deze krijgt. Maar commit access is slechts de best meetbare manifestatie van verloren invloed. Een lange-termijnontwikkelaar kent tevens alle oude argumenten die bij herhaling zijn besproken op de discussielijsten. Een nieuwe ontwikkelaar, die deze discussies niet kent, kan proberen het probleem opnieuw naar voren te brengen, wat leidt tot een verlies van geloofwaardigheid voor uw organisatie. De anderen kunnen zich dan afvragen: 'Onthouden ze dan niets?' Een nieuwe ontwikkelaar heeft ook geen politiek gevoel voor de persoonlijkheden van het project en kan niet zo snel of gemakkelijk invloed uitoefenen op de ontwikkelingsrichting als iemand die er al lang is.

Train nieuwkomers door middel van een programma van betrokkenheid onder toezicht. De nieuwe ontwikkelaar moet vanaf het allereerste begin rechtstreeks in contact staan met de openbare ontwikkelgemeenschap en beginnen met bug fixes en opschoontaken. Zo kan hij de basis van de code leren en een reputatie verwerven in de gemeenschap, maar geen langdurige ontwerpdiscussies aanzwengelen. Gedurende die tijd moeten er één of meer ervaren ontwikkelaars beschikbaar zijn om vragen te beantwoorden. Zij moeten elk bericht lezen dat de nieuwkomer op

de ontwikkelingslijsten plaatst, ook als deze in threads staan waaraan de ervaren ontwikkelaars normaliter geen aandacht besteden. Hierdoor kan de groep hopelijk zandbanken ontdekken voordat de nieuwkomer vastloopt. Afzonderlijke coaching en aanwijzingen achter de schermen kunnen ook veel helpen, vooral als de nieuwkomer niet gewend is aan grootschalige gelijktijdige beoordeling van zijn code door collega's.

Wanneer CollabNet een nieuwe ontwikkelaar inhuurt voor werk aan Subversion, gaan we samen om tafel zitten en kiezen we een aantal openstaande bugs waar de nieuwe persoon zijn tanden in kan zetten. We bespreken het technische kader van de oplossingen en benoemen dan ten minste één ervaren ontwikkelaar voor de (openbare) beoordeling van de patch die de nieuwe ontwikkelaar (eveneens openbaar) post. Doorgaans kijken we niet eens naar de patch voordat de hoofdontwikkelingslijst hiernaar kijkt, hoewel we dat wel zouden kunnen als daar reden voor was. Het belangrijkste is dat de nieuwe ontwikkelaar het proces van openbare beoordeling doorloopt, waarbij hij de basis van de code leert terwijl hij tegelijkertijd gewend raakt aan het krijgen van kritiek van wildvreemden. Maar we proberen de timing zodanig te coördineren dat onze eigen beoordeling vlak na het plaatsen van de patch komt. Op die manier ziet de lijst onze beoordeling als eerste, wat kan helpen de toon te zetten voor de andere beoordelingen. Het draagt tevens bij aan het idee dat deze nieuwe persoon serieus genomen moet worden: als anderen zien dat wij de tijd nemen om gedetailleerde beoordelingen te geven, met grondige uitleg en zo nodig verwijzingen naar de archieven, zullen ze beseffen dat het hier om een vorm van training gaat en dat dit waarschijnlijk duidt op een lange-termijninvestering. Dit kan er ook voor zorgen dat ze positiever naar die ontwikkelaar kijken, althans in zoverre dat ze wat extra tijd investeren in het beantwoorden van vragen en het beoordelen van patches.

5.3 KOM OVER ALS INDIVIDUEN, NIET ALS EEN BLOK

Uw ontwikkelaars moeten ernaar streven om in de publieke forums van het project te verschijnen als afzonderlijke deelnemers en niet als een monolithische bedrijfsgroep. Dit is niet omdat het fenomeen monolithische bedrijfsgroep negatieve gedachten oproept (nou ja, misschien ook wel, maar daar gaat dit boek niet over). Het is meer omdat afzonderlijke personen de enige entiteit zijn waar open sourceprojecten structureel mee om kunnen gaan. Een afzonderlijke bijdrager kan discussies voeren, patches indienen, geloofwaardigheid verwerven, een stem uitbrengen enz. Een bedrijf kan dat niet.

Doordat u zich op gedecentraliseerde wijze gedraagt, voorkomt u tevens dat centralisatie van de oppositie wordt bevorderd. Laat uw ontwikkelaars het met elkaar oneens zijn op de mailinglijsten. Stimuleer ze om elkaars code zo vaak en zo openbaar te beoordelen als ze dat met die van anderen zouden doen. Zorg ervoor dat ze niet altijd als een blok stemmen, omdat anderen anders het gevoel kunnen krijgen dat er, op basis van algemene principes, sprake zou moeten zijn van een georganiseerde inspanning om hen onder controle te houden.

Er is een verschil tussen echt gedecentraliseerd zijn en er alleen maar naar streven om gedecentraliseerd te lijken. Onder bepaalde omstandigheden kan het vrij nuttig zijn om uw ontwikkelaars gezamenlijk te laten optreden, en ze moeten erop voorbereid zijn zich om zo nodig achter de schermen te bundelen. Bij het doen van een voorstel kan het bijvoorbeeld helpen om verschillende mensen in een vroeg stadium hiermee te laten instemmen, zodat de indruk van een toenemende consensus wordt gewekt. Anderen zullen het idee krijgen dat het voorstel potentieel heeft en dat ze dat potentieel tegenwerken als ze er bezwaar tegen maken. Daardoor zullen mensen alleen bezwaar maken als ze er een goede reden voor hebben. Er is niets mis mee om instemming op deze wijze te organiseren, zolang bezwaren maar serieus worden genomen. Openbare manifestaties van onderhandse overeenkomsten zijn niet minder oprecht alleen omdat ze van tevoren zijn gecoördineerd. Ze zijn onschadelijk zolang ze maar niet worden gebruikt om op vooringenomen wijze tegenargumenten de nek om te draaien. Het doel ervan is louter om het type mensen af te remmen dat alleen maar bezwaar maakt omwille van het bezwaar maken. Zie het gedeelte 'Hoe makkelijker het onderwerp, des langer de discussie' in Hoofdstuk 6, *Communicatie,* voor meer hierover.

5.4 WEES OPEN OVER UW BEWEEGREDENEN

Wees zo open over de doelen van uw organisatie als mogelijk is, zonder bedrijfsgeheimen in gevaar te brengen. Als u wilt dat het project een bepaalde functie krijgt omdat uw klanten hier bijvoorbeeld om schreeuwen, zeg dit dan openlijk op de mailinglijst. Als de klanten anoniem willen blijven, wat soms gebeurt, vraag hen dan ten minste of ze mogen worden gebruikt als anonieme voorbeelden. Hoe meer de openbare ontwikkelgemeenschap weet over *waarom* u wilt wat u wilt, des te makkelijker ze aanvaarden wat u voorstelt.

Dit gaat in tegen het adagium – dat men zich zo makkelijk eigen maakt en maar zelden loslaat – dat kennis macht is en dat hoe meer anderen weten over uw doelen, des te meer macht ze over u hebben. Dat adagium geldt hier echter niet. Door de functie (of de bugfix, of wat dan ook) openbaar te promoten, *hebt* u uw kaarten al op tafel gelegd. De enige vraag is nu of u erin slaagt om de gemeenschap zo te sturen dat zij uw doel deelt. Als u alleen maar zegt dat u het wilt, maar geen concrete voorbeelden kunt geven over het waarom, is uw argument zwak en zullen mensen een verborgen agenda beginnen te vermoeden. Maar al een paar echte scenario's waarin wordt aangetoond waarom de voorgestelde functie van belang is, kunnen van grote invloed zijn op de discussie.

Om de reden hiervoor te begrijpen, moet u het alternatief overwegen. Maar al te vaak zijn discussies over nieuwe functies of nieuwe richtingen lang en vermoeiend. De argumenten die mensen naar voren brengen, zijn vaak beperkt tot 'Ik zelf wil X', of het zeer populaire 'In mijn jarenlange ervaring als softwareontwerper is X van groot belang gebleken voor gebruikers / nutteloze franje waar niemand op zit te wachten'. Zoals kan worden verwacht, worden deze discussies niet korter of gematigder door het gebrek aan echte gebruiksgegevens, maar kunnen ze daarentegen steeds verder afdrijven van het vastleggen van feitelijke gebruikservaringen. Zonder

een tegenkracht wordt het eindresultaat waarschijnlijk bepaald door degene die zich het beste kan uitdrukken, of door de meest volhardende of de meest ervaren persoon.

Als organisatie met de beschikking over veel klantgegevens kunt u zo'n tegenkracht bieden. U kunt een kanaal zijn voor informatie die anders geen middel heeft om de ontwikkelgemeenschap te bereiken. Het feit dat die informatie uw wensen ondersteunt, is helemaal niets om u voor te schamen. De meeste ontwikkelaars hebben zelf geen brede ervaring met hoe de software wordt gebruikt die zij schrijven. Elke ontwikkelaar gebruikt de software op zijn of haar eigen idiosyncratische manier. Wat betreft andere gebruikspatronen vertrouwt hij op intuïtie en giswerk en is zich hier, diep van binnen, van bewust. Door geloofwaardige gegevens te verstrekken over een groot aantal gebruikers geeft u de openbare ontwikkelgemeenschap een soort zuurstof. Zolang u het op de juiste wijze presenteert, zullen zij dit enthousiast verwelkomen en zal de zaak in de richting bewegen die u nastreeft.

De sleutel hiertoe is uiteraard juiste presentatie. Het is niet goed om erop te staan dat uw oplossing wordt geïmplementeerd alleen omdat u te maken hebt met een groot aantal gebruikers en omdat zij een bepaalde functie nodig hebben (of denken te hebben). In plaats daarvan richt u uw eerste posts op het probleem, in plaats van op één bepaalde oplossing. Beschrijf uitvoerig welke ervaringen uw klanten hebben, geef zoveel analyse als u beschikbaar hebt en zoveel redelijke oplossingen als u maar kunt bedenken. Als mensen beginnen te speculeren over de effectiviteit van verschillende oplossingen, kunt u blijven putten uit uw gegevens om datgene wat zij zeggen te ondersteunen of te weerleggen. U kunt de hele tijd al één bepaalde oplossing in gedachten hebben, maar leg hier in het begin niet een bijzondere nadruk op. Dit is geen bedrog, maar eenvoudigweg standaardgedrag van een 'eerlijke makelaar'. Uw ware doel is immers het oplossen van het probleem. Een oplossing is slechts een middel voor dat doel. Als de oplossing die uw voorkeur heeft inderdaad de beste is, zullen andere ontwikkelaars dat uiteindelijk zelf erkennen en gaan zij daar vervolgens uit eigen vrije wil achter staan. Dat is veel beter dan wanneer u hen koeioneert om deze te implementeren. (Er bestaat ook nog de mogelijkheid dat zij een betere oplossing bedenken.)

Dat wil niet zeggen dat u nooit een specifieke oplossing kan aanmoedigen. Maar u moet het geduld hebben om de analyse die u intern reeds hebt gemaakt, overgenomen te zien worden op de openbare ontwikkelingslijsten. Plaats geen bericht zoals: 'Ja, daar hebben we het hier al over gehad, maar dat werkt niet om redenen A, B en C. Als je er goed naar kijkt, is de enige manier om dit op te lossen …'. Het probleem is niet zo zeer dat dit arrogant klinkt, maar dat het de indruk wekt dat u *inmiddels al* de nodige (mensen zullen uitgaan van veel) analytische energie en aandacht achter gesloten deuren aan het probleem hebt gewijd. Dit wekt de indruk dat er inspanningen zijn geleverd en misschien wel besluiten zijn genomen waarvan het publiek niet op de hoogte is. Dat is een beproefd recept voor verontwaardiging.

U weet uiteraard hoeveel inspanning u intern aan het probleem hebt gewijd. Die kennis is, in zekere zin, een nadeel. Het plaatst uw ontwikkelaars in een iets andere mentale omgeving dan alle anderen op de mailinglijsten, waardoor hun vermogen

wordt beperkt om dingen te bekijken vanuit het standpunt van degenen die nog niet zo over het probleem hebben nagedacht. Hoe eerder u iedereen over dingen op dezelfde manier aan het denken kunt zetten als u dat doet, des te kleiner dit afstandseffect zal zijn. Deze logica geldt niet alleen voor afzonderlijke technische situaties, maar ook voor de bredere taak om uw doelen zo duidelijk mogelijk te maken. Het onbekende zorgt altijd voor meer instabiliteit dan het bekende. Als mensen begrijpen waarom u wilt wat u wilt, vinden ze het prettig om met u te praten, ook wanneer ze het er niet mee eens zijn. Als ze er niet achter kunnen komen waarom u bepaalde dingen doet, zullen ze, althans een deel van de tijd, uitgaan van het ergste.

U kunt natuurlijk niet alles publiceren en dat verwachten mensen ook niet. Alle organisaties hebben geheimen. Misschien hebben organisaties met een winstoogmerk er meer, maar ook non-profitorganisaties hebben ze. Als u moet pleiten voor een bepaalde koers, maar niets bekend kunt maken over het waarom, geef dan gewoon de beste argumenten die u met die beperking kunt geven en aanvaard het feit dat u niet zoveel invloed op de discussie hebt als u wilt. Dit is een van de compromissen die u sluit om over een ontwikkelgemeenschap te beschikken die niet op uw loonlijst staat.

5.5 MET GELD KUN JE GEEN LIEFDE KOPEN

Als u een betaalde ontwikkelaar bij een project bent, bepaal dan al vroeg richtlijnen over wat u met dat geld wel en niet kunt kopen. Dit betekent niet dat u tweemaal per dag berichten moet plaatsen op de mailinglijsten waarin u uw nobele en onomkoopbare aard benadrukt. Het betekent alleen dat u moet uitkijken naar mogelijkheden om de spanningen te verminderen die door geld *kunnen* ontstaan. U hoeft er niet vanaf het begin vanuit te gaan dat er spanningen zijn. U moet laten zien dat u zich ervan bewust bent dat deze zouden kunnen ontstaan.

Een mooi voorbeeld hiervan kwam naar voren bij het Subversion-project. Subversion werd in 2000 gestart door CollabNet, dat vanaf het begin de hoofdfinancier van het project is geweest en de salarissen van verschillende ontwikkelaars betaalde (disclaimer: ik ben een van hen). Vlak nadat het project was begonnen, namen we ter ondersteuning nog een ontwikkelaar in dienst, Mike Pilaton. Er was toen al begonnen met de codering. Hoewel Subversion zich nog steeds in een pril stadium bevond, beschikte het reeds een ontwikkelaarsgemeenschap met een reeks basisregels.

De komst van Mike leidde tot een interessante vraag. Subversion had al een beleid over hoe een nieuwe ontwikkelaar commit access krijgt. Eerst zet hij enkele patches op de ontwikkelingsmailinglijst. Nadat er genoeg patches langs zijn gekomen op grond waarvan de andere committers kunnen zien dat de nieuwe bijdrager weet wat hij doet, stelt iemand voor dat hij rechtstreeks mag committen (dat voorstel is privé, zoals beschreven in het gedeelte 'Committers'). Ervan uitgaande dat de committers hiermee instemmen, mailt een van hen de nieuwe ontwikkelaar en biedt hem rechtstreekse commit access tot de repository van het project.

CollabNet had Mike specifiek ingehuurd om aan Subversion te werken. Onder degenen die hem al kenden, bestond geen twijfel over zijn coderingsvaardigheden of zijn bereidheid om aan het project te werken. Bovendien hadden de vrijwillige ontwikkelaars een zeer goede verstandhouding met de medewerkers van CollabNet en zouden ze waarschijnlijk geen bezwaar hebben gemaakt als wij Mike op de dag dat hij werd aangesteld meteen commit access hadden gegeven. Maar we wisten dat we dan een precedent zouden scheppen. Als we Mike middels een fiat commit access hadden gegeven, hadden we gezegd dat CollabNet het recht had om projectrichtlijnen te negeren, eenvoudigweg omdat het bedrijf de hoofdfinancier is. Hoewel de schade hiervan niet noodzakelijkerwijs onmiddellijk duidelijk zou zijn, zou dit geleidelijk resulteren in het feit dat de onbetaalde ontwikkelaars het gevoel zouden krijgen dat hun rechten waren ontnomen: andere mensen moeten hun commit access verdienen en CollabNet koopt het gewoon.

Mike stemde er dus mee in om zijn tewerkstelling bij CollabNet net zo te beginnen als iedere andere vrijwillige ontwikkelaar: zonder commit access. Hij stuurde patches naar de openbare mailinglijst, waar ze door iedereen konden worden – en werden – beoordeeld. We zeiden tevens op de lijst dat we dit expres op deze manier deden, zodat het absoluut duidelijk was. Na een paar zeer actieve weken van Mike droeg iemand (ik weet niet meer of dit wel of niet een CollabNet-ontwikkelaar was) hem voor commit access voor en werd hij geaccepteerd, zoals wij al hadden verwacht.

Dit soort consistentie geeft je een geloofwaardigheid die niet voor geld te koop is. En geloofwaardigheid is een waardevolle munteenheid in technische discussies: het zorgt voor immuniteit tegen vragen achteraf over iemands beweegredenen. In de hitte van de discussie zoeken mensen soms naar niet-technische manieren om de strijd te winnen. De hoofdfinancier van het project is, vanwege zijn grote betrokkenheid en duidelijke bezorgdheid over de richtingen die het project op gaat, een groter doelwit dan anderen. Door alle projectrichtlijnen van meet af aan nauwgezet na te leven, maakt de financier zichzelf niet belangrijker dan de rest.

(Zie ook de blog van Danese Cooper op http://blogs.sun.com/roller/page/DaneseCooper/20040916 voor een soortgelijk verhaal over commit access. Cooper was toen de 'open source-diva' – volgens mij was dat haar officiële titel – van Sun Microsystem en in de blog beschrijft ze hoe de Tomcat-ontwikkelaarsgemeenschap Sun zover kreeg dat het zijn eigen ontwikkelaars hield aan dezelfde normen voor commit access als de ontwikkelaars die niet van Sun waren.)

De noodzaak dat de financiers zich aan dezelfde spelregels houden als alle anderen betekent dat het bestuursmodel van 'Vriendelijke dictators' (zie het gedeelte 'Vriendelijke dictators' in Hoofdstuk 4, *Sociale en politieke infrastructuur*) iets moeilijker te bewerkstelligen is als er sprake is van financiering, vooral als de dictator voor de hoofdfinancier werkt. Omdat een dictatuur weinig regels kent, is het voor de financier moeilijk te bewijzen dat hij zich houdt aan de normen van de gemeenschap, zelfs als dat wel het geval is. Het is beslist niet onmogelijk, maar je hebt er wel een projectleider voor nodig die dingen kan bekijken vanuit het standpunt van de externe ontwikkelaars en dat van de financier, en die dienovereenkomstig handelt. Ook dan is het waarschijnlijk een goed idee om een voorstel voor een niet-dictatoriaal bestuur achter de hand te hebben, dat naar voren kan worden gebracht op het moment dat er aanwijzingen zijn van wijdverbreide ontevredenheid in de gemeenschap.

5.6 AANBESTEDING

Met aanbesteed werk moet bij open source-softwareprojecten zorgvuldig worden omgesprongen. Idealiter wilt u dat het werk van een aannemer wordt aanvaard door de gemeenschap en wordt opgenomen in de openbare distributie. In theorie maakt het niet uit wie de aannemer is, zolang hij zijn werk maar goed doet en voldoet aan de projectrichtlijnen. Theorie en praktijk stemmen soms ook met elkaar overeen: een wildvreemde die met een goede patch komt *is* doorgaans in staat om deze in de software te krijgen. Het probleem is dat het erg moeilijk is om als echte wildvreemde een goede patch te produceren voor een niet-triviale verbetering of nieuwe functie. Deze moet eerst besproken worden met de rest van het project. De duur van die discussie kan niet exact worden voorspeld. Als de aannemer per uur wordt betaald, kan dat ertoe leiden dat u meer betaalt dan u verwachtte. Als hij een forfaitair bedrag krijgt, kan hij uiteindelijk meer werk doen dan hij zich kan veroorloven.

Er zijn twee manieren om dit te omzeilen. De manier die de voorkeur verdient, is om op basis van eerdere ervaringen een gefundeerde schatting te doen over de lengte van het discussieproces, om een foutmarge in te bouwen en om het contract daarop te baseren. Het helpt ook om het probleem onder te verdelen in zo veel mogelijk kleine, afzonderlijke porties om de voorspelbaarheid van elk portie te vergroten. De andere manier is om alleen aan te besteden voor levering van een patch en om de aanvaarding van de patch in het openbare project als een afzonderlijke kwestie te behandelen. Dan wordt het veel gemakkelijker om het contract op te stellen, maar blijft u wel zitten met de last om een particuliere patch te behouden zolang u van de software afhankelijk bent, of ten minste zolang u nodig hebt om die patch of equivalente functionaliteit in de hoofdlijn te krijgen. Uiteraard is het zo, zelfs met de voorkeursmanier, dat niet in het contract zelf kan worden bepaald dat de patch daadwerkelijk in de code wordt opgenomen, omdat dat zou betekenen dat je iets verkoopt wat niet te koop is. (Wat als de rest van het project onverwacht besluit om de functie niet te ondersteunen?) Het contract kan echter een *bonafide* inspanning vereisen om de verandering aanvaard te krijgen door de gemeenschap en dat deze in de repository wordt geplaatst als de gemeenschap daarmee instemt. Als het project bijvoorbeeld geschreven normen heeft voor codewijzigingen, kan het contract naar die normen verwijzen en specificeren dat het werk daaraan moet voldoen. In de praktijk werkt dit doorgaans zoals iedereen hoopt.

De beste tactiek voor een succesvolle aanbesteding is om een van de ontwikkelaars van het project – bij voorkeur een committer – in te huren als aannemer. Dit kan lijken op een vorm van invloed kopen; en dat is het eigenlijk ook. Maar het is niet zo corrupt als het lijkt. De invloed van een ontwikkelaar op het project hangt voornamelijk af van de kwaliteit van zijn code en zijn interactie met andere ontwikkelaars.

Het feit dat hij een contract heeft om bepaalde dingen te doen, verhoogt zijn status op geen enkele wijze en verlaagt deze evenmin, hoewel mensen wel nauwlettender naar hem kunnen gaan kijken. De meeste ontwikkelaars zetten hun langetermijnpositie in het project niet op het spel door een ongeschikte of breed afgekeurde nieuwe functie te ondersteunen. Een deel van wat u krijgt, of zou moeten krijgen, door het inhuren van zo'n aannemer is in feite advies over welke soorten veranderingen waarschijnlijk worden geaccepteerd door de gemeenschap. Er komt tevens een kleine verschuiving in de prioriteiten van het project. Omdat prioriteiten stellen slechts een kwestie is van wie tijd heeft om aan wat te werken, zorgt u ervoor dat u, wanneer u voor iemands tijd betaalt, hun werk een beetje hoger plaatst in de prioriteitenrangorde. Dit is een algemeen bekend feit bij ervaren open source-ontwikkelaars en ten minste een aantal van hen zal aandacht besteden aan het werk van de aannemer, eenvoudigweg omdat het erop lijkt dat het *wordt gedaan* en ze dus willen helpen om het goed te doen. Misschien schrijven ze geen code, maar ze zullen wel het ontwerp bespreken en de code beoordelen, wat beide erg nuttig kan zijn. Om al deze redenen kan de aannemer het beste iemand zijn die reeds bij het project betrokken is.

Dit roept meteen twee vragen op: Moeten contracten altijd onderhands zijn? En als ze dit niet zijn, moet u zich dan zorgen maken over het creëren van spanningen in de gemeenschap door het feit dat u met sommige ontwikkelaars een contract hebt gesloten en met andere niet?

Het is het beste om, indien mogelijk, open kaart te spelen over contracten. Anders kan het gedrag van de aannemer op anderen in de gemeenschap vreemd overkomen. Misschien geeft hij opeens onverklaarbaar veel prioriteit aan functies waarvoor hij in het verleden nooit interesse heeft getoond. Als mensen hem vragen waarom hij deze nu wel wil, hoe kan hij dan overtuigend antwoorden als hij niet mag spreken over het feit dat hij is ingehuurd om deze te schrijven?

Tegelijkertijd moeten u noch de aannemer zich gedragen alsof anderen uw regeling als iets belangrijks moeten zien. Te vaak heb ik aannemers over een ontwikkelingslijst heen zien walsen met een houding alsof hun berichten serieuzer moeten worden genomen, eenvoudigweg omdat zij betaald worden. Een dergelijke houding maakt de rest van het project duidelijk dat de aannemer het feit van het contract – in tegenstelling tot de code die *voortvloeit* uit het contract – het belangrijkste vindt. Maar vanuit het standpunt van de andere ontwikkelaars is alleen de code van belang. De aandacht moet te allen tijde gericht blijven op technische kwesties en niet op informatie over wie wie betaalt. Een van de ontwikkelaars in de Subversion-gemeenschap gaat bijvoorbeeld op een bijzonder fraaie manier om met aanbesteding. Tijdens de bespreking van zijn codewijzigingen in IRC merkt hij terloops op (vaak met een privéopmerking, een IRC *privmsg*, aan een van de andere committers) dat hij wordt betaald voor zijn werk aan deze specifieke bug of functie. Maar hij geeft ook consistent de indruk dat hij toch al graag aan die wijziging wilde werken en dat hij blij is dat het geld het hem mogelijk maakt om dat te doen. Hij kan al dan niet de identiteit van zijn klant onthullen, maar hij weidt in elk geval niet uit over het contract. Zijn opmerkingen daarover zijn slechts een extra in een verder technische discussie over hoe iets gedaan moet worden.

Dat voorbeeld laat nog een reden zien waarom het goed is om open kaart te spelen over contracten. Er kunnen meer organisaties zijn die contracten voor een bepaald open source-project sponsoren en als iedereen weet wat de anderen proberen te doen, kunnen ze hun krachten wellicht bundelen. In het bovenstaande geval is de grootste financier van het project (CollabNet) op geen enkele wijze betrokken bij deze stukwerkcontracten, maar in de wetenschap dat iemand anders bepaalde fixes van bugs sponsort, kan CollabNet zijn medewerkers inzetten voor andere bugs, wat leidt tot een grotere efficiëntie voor het hele project.

Zullen andere ontwikkelaars ontstemd zijn dat sommigen worden betaald om aan het project te werken? Doorgaans niet. Vooral wanneer degenen die betaald worden toch al gevestigde en gerespecteerde leden van de gemeenschap zijn. Niemand verwacht dat aanbestedingswerk gelijk wordt verdeeld onder alle committers. Mensen begrijpen het belang van langetermijnrelaties: de onzekerheden bij aanbesteding zijn zodanig dat je, wanneer je eenmaal iemand vindt met wie je betrouwbaar kunt werken, liever niet overstapt naar een andere persoon alleen maar omwille van de gelijkheid. Bekijk het op deze manier: de eerste keer dat u iemand inhuurt, komen er geen klachten omdat u overduidelijk *iemand* moest kiezen; het is niet uw schuld dat u niet iedereen in kunt huren. Later, wanneer u dezelfde persoon nogmaals inhuurt, is dat alleen maar gezond verstand: u kent hem al, de laatste keer ging het goed, dus waarom zou u onnodige risico's nemen? Daarom is het volkomen normaal om één of twee ingehuurde mensen in de gemeenschap te hebben, in plaats van het werk evenredig te verdelen.

Beoordeling en aanvaarding van wijzigingen

De gemeenschap is nog steeds belangrijk voor het welslagen van aanbestedingswerk. Hun betrokkenheid bij het ontwerp- en beoordelingsproces voor grote veranderingen kan geen overweging achteraf zijn. Deze moet worden gezien als deel van het werk en moet volledig worden aanvaard door de aannemer. Zie de beoordeling van de gemeenschap niet als een hindernis die overwonnen moet worden. Zie het als een gratis ontwerpraad en afdeling kwaliteitsgarantie. Het is een voordeel om agressief achtervolgd te worden en niet slechts te worden verdragen.

Casestudy: het protocol voor CVS-wachtwoordauthenticatie

In 1995 was ik één helft van een partnerschap dat ondersteuning en verbeteringen leverde aan CVS (het Concurrent Versions System; zie http://www.cvshome.org/). Mijn partner Jim en ik waren, informeel, op dat moment degenen die het onderhoud van CVS deden. Maar we dachten nooit goed na over onze relatie met de bestaande, veelal vrijwillige CVS-ontwikkelaarsgemeenschap. We gingen er gewoon vanuit dat zij patches zouden sturen en dat wij ze zouden toepassen; en zo werkte het ook eigenlijk wel.

Toentertijd kon CVS in een netwerk alleen worden gedaan via een inlogprogramma op afstand, zoals `rsh`. Hetzelfde wachtwoord gebruiken voor CVS-toegang en om in te loggen vormde een duidelijk beveiligingsrisico. Dit schrikte veel organisaties af om deze methode toe te passen. Een grote investeringsbank huurde ons in om een nieuw authenticatiemechanisme toe te voegen, zodat ze op een veilige manier CVS in een netwerk konden gebruiken vanuit hun kantoren op afstand.

Jim en ik namen de opdracht aan en gingen aan het werk om het nieuwe authenticatiesysteem te ontwerpen. Waar we mee kwamen, was vrij eenvoudig. De Verenigde Staten hadden toentertijd exportcontrole op cryptografische code, dus de klant begreep dat we geen zware authenticatie konden implementeren. Omdat we echter niet ervaren waren in het ontwerpen van dergelijke protocollen begingen we toch een paar blunders die een deskundige had weten te voorkomen. Deze fouten hadden gemakkelijk kunnen worden opgemerkt als we de tijd hadden genomen om een voorstel op te schrijven en dit ter beoordeling voor te leggen aan de andere ontwikkelaars. Maar dat hebben we nooit gedaan, omdat we er simpelweg niet aan gedacht hadden om de ontwikkelingslijst als een hulpmiddel te gebruiken. We wisten dat men waarschijnlijk sowieso zou accepteren wat wij committen en, omdat we niet wisten wat we niet wisten, namen we niet de moeite om het werk op een zichtbare manier te doen, bijvoorbeeld door vaak patches te posten, kleine, gemakkelijk te behappen commits aan een bepaalde branch te doen enz. Het authenticatieprotocol dat daaruit voortvloeide, was niet erg goed en was uiteraard, toen het eenmaal gevestigd was, moeilijk te verbeteren vanwege compatibiliteitskwesties.

De kern van het probleem was geen gebrek aan ervaring. We hadden makkelijk te weten kunnen komen wat we hadden moeten weten. Het probleem was onze houding ten opzichte van de vrijwillige ontwikkelaarsgemeenschap. Wij beschouwden aanvaarding van veranderingen als een hindernis die overwonnen moest worden en niet als een proces waarmee de kwaliteit van de veranderingen verbeterd kon worden. Omdat we ervan uitgingen dat vrijwel alles wat we deden, aanvaard zou worden (zoals gebeurde), deden we niet echt ons best om anderen erbij te betrekken.

Als u een aannemer kiest, wilt u voor het werk natuurlijk iemand met de juiste technische vaardigheden en ervaring. Maar het is ook van belang om iemand te kiezen met een staat van dienst op het gebied van constructieve interactie met de andere ontwikkelaars in de gemeenschap. Op die manier krijgt u meer dan slechts één persoon. U krijgt een agent die kan putten uit een netwerk van ervaring, zodat u er zeker van bent dat het werk op een solide en onderhoudsvriendelijke manier wordt gedaan.

5.7 FINANCIERING VAN NIET-PROGRAMMERINGS-ACTIVITEITEN

Programmering is slechts een deel van het werk dat wordt gedaan bij een open source-project. Vanuit het standpunt van de vrijwilligers van het project is dit het meest zichtbare en aantrekkelijke deel. Dat betekent jammer genoeg dat andere activiteiten, zoals documentatie, formeel testen enz., soms worden verwaarloosd, althans in vergelijking met de hoeveelheid aandacht die daar bij gepatenteerde software vaak aan wordt besteed. Bedrijven kunnen dit soms compenseren door een deel van hun interne softwareontwikkelingsinfrastructuur in te zetten voor open source-projecten.

De sleutel om dit succesvol te doen is om een vertaalslag te maken tussen de interne processen in het bedrijf en die van de openbare ontwikkelaarsgemeenschap. Deze vertaalslag kost moeite. Vaak sluiten beide processen niet goed op elkaar aan en kunnen de verschillen alleen overbrugd worden door menselijk ingrijpen. Het bedrijf kan bijvoorbeeld gebruik maken van een andere bug tracker dan het openbare project. Ook al gebruiken ze dezelfde opsporingssoftware, dan nog zullen de gegevens die daarin zijn opgeslagen zeer verschillend zijn, omdat de behoeften voor bugopsporing van een bedrijf sterk verschillen van die van een open source-softwaregemeenschap. Een stuk informatie dat in één bug tracker begint, moet mogelijk worden weergegeven in de andere, waarbij vertrouwelijke delen worden verwijderd of juist worden toegevoegd.

De onderstaande onderdelen gaan over het bouwen van dergelijke bruggen en het onderhoud ervan. Het eindresultaat moet zijn dat het open source-project soepeler verloopt, dat de gemeenschap ziet welke investering in middelen het bedrijf heeft gepleegd en toch niet het idee heeft dat het bedrijf op ongepaste wijze de richting manipuleert ten gunste van zijn eigen doelstellingen.

Kwaliteitsgarantie (oftewel professioneel testen)

Bij de ontwikkeling van gepatenteerde software is het gebruikelijk dat teams van mensen zich alleen richten op kwaliteitsgarantie: opsporen van bugs, testen van prestatie en schaalbaarheid, controle van interface en documentatie enz. Als regel worden deze activiteiten niet zo sterk nagestreefd door de vrijwillige gemeenschap van een open source-softwareproject. Dit komt deels omdat het moeilijk is om vrijwilligers te vinden voor onaantrekkelijk werk zoals testen, deels omdat mensen de neiging hebben ervan uit te gaan dat een grote gebruikersgemeenschap het project een goede testdekking geeft en, in het geval van het testen van prestaties en schaalbaarheid, omdat vrijwilligers vaak toch geen toegang hebben tot de benodigde hardware.

De veronderstelling dat het hebben van veel gebruikers gelijk staat aan het hebben van veel testers is niet geheel ongegrond. Het heeft zeker weinig zin om testers aan te stellen voor basisfunctionaliteiten in gemeenschappelijke omgevingen Bugs worden daar bij de normale gang van zaken snel gevonden door gebruikers. Maar omdat gebruikers alleen maar proberen om werk gedaan te krijgen, gaan ze niet bewust op verkenning in niet in kaart gebrachte randgevallen in de functionaliteit van het programma en vinden ze bepaalde categorieën bugs waarschijnlijk niet. En als ze een bug ontdekken die makkelijk te omzeilen is, implementeren ze bovendien vaak zonder het te zeggen de programmaomleiding zonder de moeite te nemen om de fout te melden. Wat veel verraderlijker is, is dat de gebruikspatronen van uw klanten (de mensen die de motor zijn van *uw* belang in de software) statistisch grote verschillen kunnen vertonen met die van de ‘gemiddelde gebruiker op straat’.

Een professioneel testteam kan dit soort bugs aan het licht brengen en kan dit net zo makkelijk doen bij open source-software als bij gepatenteerde software. De uitdaging is om de resultaten van het testteam op een nuttige wijze terug te koppelen naar het publiek. Interne testafdelingen hebben doorgaans hun eigen manier om testresultaten te rapporteren, waarbij bedrijfsspecifiek jargon wordt gebruikt of gespecialiseerde kennis over bepaalde klanten en hun datasets vereist is. Deze rapporten zijn niet geschikt voor de openbare bug tracker, zowel vanwege hun vorm als vanwege vertrouwelijkheidskwesties. Zelfs als de interne foutopsporingssoftware

van uw bedrijf dezelfde is als degene die wordt gebruikt door het openbare project, moet het management misschien bedrijfsspecifieke opmerkingen en metadatawijzigingen in de problemen maken (bijvoorbeeld om de interne prioriteit van een kwestie hoger te maken of om de oplossing daarvan te plannen voor een bepaalde klant). Doorgaans zijn dergelijke notities vertrouwelijk. Soms krijgt zelfs de klant ze niet te zien. Maar ook als ze niet vertrouwelijk zijn, zijn ze niet van belang voor het openbare project en moet het publiek er daarom niet door worden afgeleid.

Maar het basale bugrapport zelf is *wel* van belang voor het publiek. Een foutenrapportage van uw testafdeling is op bepaalde manieren namelijk waardevoller dan eentje die wordt ontvangen van gebruikers in extenso, omdat de testafdeling op zoek gaat naar dingen die andere gebruikers niet zoeken. Aangezien u die bepaalde foutrapportage waarschijnlijk niet uit een andere bron zult krijgen, wilt u deze beslist behouden en beschikbaar stellen voor het openbare project.

Hiertoe kan de afdeling kwaliteitsgarantie problemen rechtstreeks archiveren in de openbare issue tracker, als ze dat gemakkelijk vinden, of kan een tussenpersoon (meestal een van de ontwikkelaars) de interne rapporten van de testafdeling 'vertalen' in nieuwe problemen in de openbare tracker. Vertaling betekent eenvoudigweg het beschrijven van de bug op een manier die geen verwijzing bevat naar klantspecifieke informatie (het reproductierecept kan gebruik maken van klantgegevens, uiteraard mits de klant hiervoor toestemming geeft).

Het verdient de voorkeur om de afdeling kwaliteitsgarantie problemen rechtstreeks te laten archiveren in de openbare tracker. Dat zorg er op een meer directe manier voor dat het publiek waardering krijgt voor de betrokkenheid van uw bedrijf bij het project: nuttige bugrapporten vergroten de geloofwaardigheid van uw organisatie, net zoals iedere technische bijdrage dat zou doen. Het geeft ontwikkelaars tevens een rechtstreekse communicatielijn met het testteam. Als het interne team kwaliteitsgarantie bijvoorbeeld toezicht houdt op de openbare foutopsporing, kan een ontwikkelaar een fix committen voor een bug op het gebied van schaalbaarheid (waarvoor de ontwikkelaar misschien niet de middelen heeft om dit zelf te testen) en daarna een notitie over het issue toevoegen waarin het team kwaliteitsgarantie wordt gevraagd of de reparatie het gewenste effect had. Verwacht enige weerstand van sommige ontwikkelaars. Programmeurs hebben de neiging om kwaliteitsgarantie op z'n best te zien als een noodzakelijk kwaad. Het team kwaliteitsgarantie kan dit gemakkelijk overwinnen door grote bugs te vinden en begrijpelijke rapporten in te dienen. Aan de andere kant is het zo dat, als hun rapporten niet ten minste net zo goed zijn als die van de gebruikelijke gebruikersgemeenschap, het geen zin heeft om deze rechtstreeks in contact te laten staan met het ontwikkelingsteam.

Hoe dan ook, wanneer er een openbaar issue is, moet de oorspronkelijke interne issue eenvoudigweg naar de openbare issue verwijzen voor technische inhoud. Management en betaalde ontwikkelaars kunnen doorgaan met het annoteren van de interne issue met zo nodig bedrijfsspecifieke opmerkingen, maar kunnen de openbare issue gebruiken voor informatie die voor iedereen beschikbaar zou moeten zijn.

Wanneer u dit proces ingaat, moet u extra overhead verwachten. Het onderhouden van twee issues voor één bug is, uiteraard, meer werk dan het onderhouden van één issue. Het voordeel is dat veel meer programmeurs het rapport zien en kunnen bijdragen aan een oplossing.

Juridisch advies en bescherming

Bedrijven, al dan niet met winstoogmerk, zijn vrijwel de enige entiteiten die aandacht besteden aan complexe juridische kwesties bij open source-software. Afzonderlijke ontwikkelaars begrijpen vaak de nuances van verschillende open source-licenties, maar hebben doorgaans de tijd noch de middelen om wetgeving op het gebied van auteursrechten, handelsmerken en octrooien gedetailleerd te volgen. Als uw bedrijf een afdeling juridische zaken heeft, kan deze een project helpen door de auteursrechtstatus van de code door te lichten en de ontwikkelaars te helpen inzicht te krijgen in mogelijke problemen op het gebied van octrooien en handelsmerken. De exacte vorm die deze hulp kan krijgen, wordt besproken in Hoofdstuk 9, *Licenties, auteursrechten en octrooien*. Het belangrijkste is om ervoor te zorgen dat de communicatie tussen de afdeling juridische zaken en de ontwikkelaarsgemeenschap, als deze er al is, plaatsvindt in een sfeer van wederzijds inzicht in de zeer verschillende werelden waar de partijen vandaan komen. Soms praten deze twee groepen langs elkaar heen, waarbij elke groep uitgaat van gebiedsspecifieke kennis die de andere niet heeft. Een goede strategie is om een tussenpersoon te hebben (meestal een ontwikkelaar of anders een jurist met technische expertise) die zo lang het nodig is de boodschappen vertaalt.

Documentatie en bruikbaarheid

Documentatie en bruikbaarheid zijn beide beroemde zwakke plekken bij open source-projecten, hoewel ik denk, althans in het geval van documentatie, dat het verschil tussen vrije en gepatenteerde software vaak wordt overdreven. Het is echter empirisch vastgesteld dat veel open source-software eersteklasonderzoek naar documentatie en bruikbaarheid ontbeert.

Als uw organisatie deze gaten in een project wil helpen dichten, kan deze waarschijnlijk het beste mensen inhuren die *niet* de gebruikelijke ontwikkelaars van het project zijn, maar die in staat zijn om productief met de ontwikkelaars samen te werken. Het niet inhuren van de gebruikelijke ontwikkelaars is om twee redenen gunstig. Allereerst hoeft u geen ontwikkelingstijd te onttrekken aan het project en ten tweede zijn degenen die het dichtst bij de software staan toch vaak de verkeerde mensen om documentatie te schrijven of onderzoek te doen naar bruikbaarheid, omdat ze de software maar moeilijk kunnen bekijken door de ogen van een buitenstaander.

Degene die aan deze problemen werkt, moet echter wel met de ontwikkelaars communiceren. Zoek mensen die technisch genoeg zijn om met het programmeringsteam te praten, maar die niet zo deskundig zijn op het gebied van de software dat ze zich niet meer in kunnen leven in de normale gebruiker.

Een gebruiker op gemiddeld niveau is waarschijnlijk de juiste persoon om goede documentatie te schrijven. Na publicatie van de eerste uitgave van dit boek kreeg ik de volgende e-mail van een open source-ontwikkelaar genaamd Dirk Reiners:

Een opmerking bij Geld: Documentatie en bruikbaarheid: toen we wat geld hadden en besloten dat een beginnershandleiding het belangrijkste was wat we nodig hadden, huurden we een gebruiker op gemiddeld niveau in om deze te schrijven. Hij had recent genoeg de inleiding bij het systeem doorlopen om zich de problemen te herinneren, maar hij had deze al overwonnen en wist dus hoe hij ze moest beschrijven. Daardoor kon hij iets schrijven waar door de kernontwikkelaars slechts wat kleine aanpassingen in hoeden worden aangebracht met betrekking tot dingen die hij niet bij het juiste eind had, maar waar nog steeds alle 'voor de hand liggende' dingen instonden die ontwikkelaars zouden hebben overgeslagen.

In zijn geval ging het nog beter, omdat hij de opdracht had gekregen om een aantal andere mensen (studenten) bekend te maken met het systeem. Daardoor kon hij de ervaring van veel mensen combineren, wat gewoon een gelukkige samenloop van omstandigheden was en in de meeste gevallen waarschijnlijk niet voorkomt.

Hosting/bandbreedte bieden
Voor een project dat geen gebruik maakt van een van de gratis standaard-hostingsites (zie het gedeelte 'Canned hosting' in Hoofdstuk 3, *Technische infrastructuur*), kan het bieden van een server- en netwerkverbinding – en zeer belangrijk: hulp van systeembeheerders – veel hulp bieden. Ook als dit alles is wat uw organisatie voor het project doet, kan het een vrij doeltreffende manier zijn om goed pr-karma te verkrijgen, hoewel het geen invloed op de richting van het project met zich meebrengt.

U kunt waarschijnlijk een banner of een blijk van waardering op de homepage van het project verwachten, waarin uw bedrijf wordt bedankt voor het leveren van hosting. Als u de hosting zodanig opzet dat het webadres van het project onder uw domeinnaam staat, krijg u wat extra associatie via de URL. Dit zorgt ervoor dat de meeste gebruikers denken dat de software *iets* met uw bedrijf te maken heeft, ook al draagt u op geen enkele wijze bij aan de ontwikkeling. Het probleem is dat de ontwikkelaars zich eveneens bewust zijn van deze associatie en zich er niet prettig bij voelen om het project in uw domein te hebben, tenzij u meer bijdraagt dan alleen bandbreedte. Er zijn tegenwoordig immers veel plaatsen om te hosten. De gemeenschap kan uiteindelijk het idee hebben dat de geïmpliceerde misplaatsing van credit het gemak van hosting niet waard is en het project elders onderbrengen. Dus als u hosting wilt bieden, kunt u dat doen, maar plan dan om binnenkort nauwer betrokken te raken of wees op uw hoede ten aanzien van de hoeveelheid betrokkenheid u opeist.

5.8 MARKETING

Dat marketing effectief is, willen de meeste open source-ontwikkelaars waarschijnlijk niet graag toegeven. Een goede marketingcampagne *kan* ruchtbaarheid geven aan een open source-product, zelfs zodanig dat koppige programmeurs vaag positieve gedachten over de software krijgen om redenen waar ze hun vinger niet echt op kunnen leggen. Het is niet mijn taak om de dynamiek van de bewapeningswedloop van marketing in het algemeen te ontleden. Elk bedrijf dat betrokken is bij open source-software gaat uiteindelijk overwegen hoe het zichzelf, de software of zijn relatie tot de software gaat marketen. Het onderstaande advies gaat over het vermijden van algemene valkuilen bij een dergelijke inspanning. Zie ook het gedeelte 'Publiciteit' in Hoofdstuk 6, *Communicatie*.

Onthoud dat u in de gaten gehouden wordt
Om de vrijwillige ontwikkelaarsgemeenschap aan uw zijde te houden, is het van *groot* belang om niets te zeggen wat niet aantoonbaar waar is. Controleer alle beweringen zorgvuldig voordat u ze doet en geef het publiek de middelen om uw beweringen zelf te controleren. Onafhankelijke feitencontrole vormt een belangrijk deel van open source en dit geldt voor meer dan alleen de code.

Natuurlijk zal niemand bedrijven adviseren om oncontroleerbare beweringen te doen. Maar bij open source-activiteiten is er een ongebruikelijk groot aantal mensen met de expertise om beweringen te controleren; mensen die waarschijnlijk ook zeer uitgebreide internettoegang hebben en de juiste sociale contacten om hun bevindingen op schadelijke wijze te publiceren, als zij daarvoor kiezen. Wanneer Global Megacorp Chemical Industries een rivier vervuilt, is dat controleerbaar, maar alleen door getrainde wetenschappers die vervolgens kunnen worden weersproken door de wetenschappers van Global Megacorp, zodat het publiek zich afvraagt welk verhaal waar is. Aan de andere kant is uw gedrag in de open source-wereld niet alleen zichtbaar en vastgelegd, het is ook voor veel mensen gemakkelijk om dit onafhankelijk te controleren, hun eigen conclusies te trekken en deze conclusies te verspreiden via mond-tot-mondcommunicatie. Deze communicatienetwerken bestaan al. Ze zijn de essentie van hoe open source werkt en kunnen worden gebruikt om allerlei soorten informatie te verspreiden. Weerlegging is vaak moeilijk, zo niet onmogelijk, vooral als datgene wat mensen zeggen waar is.

Het is bijvoorbeeld in orde om te zeggen dat uw organisatie 'project X heeft gefinancierd' als u dat daadwerkelijk hebt gedaan. Maar noem uzelf niet de 'makers van X' als de meeste code is geschreven door buitenstaanders. Zeg omgekeerd niet dat u een sterk betrokken vrijwillige ontwikkelaarsgemeenschap hebt als iedereen uw repository kan bekijken en kan zien dat er weinig tot geen codewijzigingen afkomstig zijn van buiten uw organisatie.

Niet zo lang geleden zag ik een aankondiging van een zeer bekend computerbedrijf dat zei dat ze een belangrijk softwarepakket publiceerden onder een open sourcelicentie. Toen de aanvankelijke aankondiging uitkwam, keek ik naar hun inmiddels publieke versiebeheer-repository en zag dat deze maar drie revisies bevatte. Met andere woorden, ze hadden eerst de broncode geïmporteerd, maar sindsdien weinig gedaan. Dat is op zich niet verontrustend. Ze hadden dit immers alleen maar aangekondigd. Er was geen reden om onmiddellijk veel ontwikkelingsactiviteit te verwachten.

Iets later deden ze een andere mededeling. Dit is wat werd gezegd, waarbij de naam en het publicatienummer zijn vervangen door pseudoniemen:

Met genoegen kondigen wij aan dat, na grondig te zijn getest door de Singer-gemeenschap, Singer 5 voor Linux en Windows nu gereed is voor productiegebruik.

Uit nieuwsgierigheid over wat de gemeenschap had ontdekt tijdens het 'grondig testen', ging ik terug naar de repository om te kijken naar de recente change history. Het project zat nog steeds in revisie 3. Blijkbaar hadden ze geen *enkele* bug gevonden die het waard was om voor release te worden hersteld! Omdat ik dacht dat de resultaten van de gemeenschaptest elders moesten zijn opgeslagen, keek ik vervolgens naar de bug tracker. Er stonden precies zes issues open, waarvan er vier al enkele maanden openstonden.

Dit gaat natuurlijk ten koste van de geloofwaardigheid. Wanneer testers gedurende een bepaalde tijd zwoegen op een groot en complex stuk software vinden ze bugs. Zelfs als de fixes van die fouten niet worden opgenomen in de komende release, zou je toch nog enige versiebeheeractiviteiten verwachten als gevolg van het testproces, of ten minste een aantal nieuwe issues. Maar naar het schijnt, was er niets gebeurd tussen de aankondiging van de open source-licentie en de eerste open source-release.

Het punt is niet dat het bedrijf loog over de test door de gemeenschap. Ik heb geen idee of dit zo was. Maar ze waren zich niet bewust van hoeveel het *erop leek* dat ze logen. Aangezien de versiebeheer-repository noch de issue tracker enige indicatie gaf dat de aangevoerde grondige test had plaatsgevonden, had het bedrijf de aanspraak allereerst niet moeten maken, of had het een duidelijke link moeten verstrekken naar een tastbaar resultaat van die test ('We hebben 278 fouten gevonden; klik hier voor informatie'). Dit laatste had het iedereen mogelijk gemaakt om heel snel een idee te krijgen van de mate van gemeenschapactiviteit. In dit geval kostte het me een paar minuten om vast te stellen dat, wat deze gemeenschaptest ook inhield, deze in elk geval geen sporen had achtergelaten op een gebruikelijke plaatsen. Dat is niet veel moeite en ik weet zeker dat ik niet de enige ben die deze moeite heeft genomen.

Transparantie en controleerbaarheid zijn uiteraard eveneens een belangrijk onderdeel van geloofwaardigheid. Raadpleeg voor meer informatie het gedeelte 'Erkenning' in Hoofdstuk 8*, Het managen van vrijwilligers*.

Kraak concurrerende open source-producten niet af

Onthoud u van negatief commentaar over concurrerende open source-software. Het is geen probleem om met negatieve *feiten* te komen, oftewel gemakkelijk te bevestigen beweringen zoals je vaak ziet in goede vergelijkingstabellen. Maar negatieve typeringen van een minder strikte aard kunnen het beste worden vermeden. Hiervoor zijn twee redenen. Allereerst kunnen ze leiden tot heftige ruzies die afleiden van een productieve discussie. Ten tweede, en nog belangrijker, kunnen sommige vrijwillige ontwikkelaars van *uw* project tevens aan het concurrerende product blijken te werken. Dit is waarschijnlijker dan het op het eerste gezicht lijkt. De projecten bevinden zich al op hetzelfde terrein (daarom zijn ze concurrerend) en ontwikkelaars met expertise op dat gebied kunnen bijdragen leveren overal waar hun expertise kan worden toegepast. Ook als er geen directe ontwikkelaarsoverlapping is, is het waarschijnlijk dat de ontwikkelaars bij uw project op z'n minst ontwikkelaars bij gerelateerde projecten *kennen*. Hun vermogen om constructieve persoonlijke relaties te onderhouden, kan worden belemmerd door al te negatieve marketingboodschappen.

Het afkraken van concurrerende closed source-producten lijkt meer algemeen aanvaard te zijn in de open source-wereld, vooral wanneer die producten zijn gemaakt door Microsoft. Persoonlijk betreur ik deze tendens (hoewel er dus niets mis is met duidelijke vergelijkingen van feiten), niet zozeer omdat het onbeleefd is, maar ook omdat het voor een project gevaarlijk is om te gaan geloven in zijn eigen hype en daarbij voorbij te gaan aan de manieren waarop de concurrentie eigenlijk beter kan zijn. Over het algemeen geldt dat u uit moet kijken voor de gevolgen die marketinguitspraken kunnen hebben voor uw eigen ontwikkelaarsgemeenschap. Mensen kunnen het zo fijn vinden om te worden gesteund door marketinggeld dat ze niet langer objectief zijn over de ware sterke en zwakke punten van hun software. Het is normaal, en zelfs te verwachten, dat de ontwikkelaars van een bedrijf een bepaalde afstand van marketinguitspraken nemen, ook in openbare forums. Uiteraard moeten ze de marketingboodschap niet openlijk en rechtstreeks tegenspreken (tenzij deze feitelijk fout is, maar je hoopt dat zoiets al eerder is opgemerkt), maar ze kunnen er af en toe grapjes over maken als een manier om de rest van de ontwikkelaarsgemeenschap weer met beide benen op de grond te zetten.