2

AAN DE GANG

Het klassieke model over hoe van start te gaan met open source-softwareprojecten is van de hand van Eric Raymond, uit een inmiddels beroemd artikel over open source-processen getiteld The Cathedral and the Bazaar. Hij schreef:

Ieder goed softwareproject begint bij de persoonlijke passie van een ontwikkelaar.
(uit: http://www.catb.org/~esr/writings/cathedral-bazaar/)

Wat opvalt is dat Raymond niet zegt dat open source-projecten alleen mogelijk zijn wanneer een individu er zin in heeft. Wat hij zegt, is dat goede software het gevolg is van een programmeur die er persoonlijk belang bij heeft dat het probleem wordt opgelost. Deze persoonlijk passie is relevant voor open source-software omdat dit in de meeste gevallen de motivatie vormt om een open source-softwareproject te beginnen.

De meeste open source-softwareprojecten beginnen nog steeds zo, maar in mindere mate dan in 1997, toen Raymond zijn wijze woorden schreef. Vandaag de dag zien we dat organisaties, waaronder commerciële bedrijven, centraal beheerde open source-projecten vanaf het eerste begin op poten zetten. De eenzame programmeur, die wat code uit zijn mouw schudt om een lokaal probleem op te lossen en zich vervolgens realiseert dat het veel breder inzetbaar is, is nog steeds een bron van veel nieuwe open source-software, maar zeker niet de enige.

Raymonds insteek is echter nog steeds waardevol. Een belangrijke voorwaarde is dat de producenten van de software zelf direct belang hebben bij het succes ervan, omdat ze het zelf gebruiken. Wanneer de software niet doet wat het zou moeten doen, zal de persoon of de organisatie daar in hun dagelijkse werkzaamheden last van hebben. Van het *OpenAdapter-project* (http://www.openadapter.org/) bijvoorbeeld, dat werd opgezet door de investeringsbank Dresdner Kleinwort Wasserstein en dat een open source-framework is voor het integreren van ongelijksoortige financiële informatiesystemen, kan nauwelijks worden gezegd dat er sprake was van persoonlijke passie van de programmeurs. Maar de instelling in kwestie had die passie wel. Die passie had direct te maken met de ervaringen van de instelling en haar partners en als het project niet in zou spelen op die ervaring, zou dat zeker

gevoeld worden. Deze manier van werken resulteert in goede software, omdat de feedbacklus in de juiste richting loopt. Het programma wordt niet geschreven om aan een ander te worden verkocht, het is bedoeld om een eigen probleem op te lossen. Het wordt geschreven om iemands eigen probleem op te lossen, waarna het met anderen wordt gedeeld. Het is alsof het probleem een ziekte is, en de software het geneesmiddel waarvan de distributie gericht is op het volledig uitroeien van de epidemie.

Dit hoofdstuk gaat over de manier waarop een nieuw open source-softwareproject bij de buitenwereld kan worden geïntroduceerd. Veel van de aanbevelingen lijken echter op die van een gezondheidsorganisatie die een geneesmiddel distribueert. De doelen komen sterk overeen: U wilt duidelijk maken wat het geneesmiddel doet en ervoor zorgen dat het in de handen van de juiste mensen komt en dat degene die het gebruikt, weet hoe het moet worden toegepast. In het geval van software wilt u echter ook sommige gebruikers overhalen deel uit te gaan maken van het lopende onderzoek om het geneesmiddel te verbeteren.

De distributie van vrije software is een tweeledige taak. Er moeten gebruikers voor de software worden gevonden, maar ook ontwikkelaars. Deze twee elementen zijn niet per definitie in strijd met elkaar, maar ze maken de eerste presentatie van het project wel gecompliceerder. Sommige informatie is bruikbaar voor beide doelgroepen, andere informatie is echter alleen bruikbaar voor de één of de ander. Beide soorten informatie moeten voldoen aan het principe van de 'geschaalde' presentatie. Dat wil zeggen dat het detailniveau van de gepresenteerde informatie in iedere fase direct overeen moet komen met de hoeveelheid tijd en moeite die de lezer erin stop. Meer moeite moet altijd meer worden beloond. Wanneer er geen zeer sterk verband is tussen de twee kunnen mensen hun vertrouwen snel verliezen en willen ze er geen moeite meer voor doen.

Een logisch gevolg hiervan is dat het uiterlijk er wel degelijk toe doet. Met name programmeurs zijn niet altijd geneigd dit te geloven. Hun liefde voor inhoud boven vorm kan bijna worden gezien als een vorm van beroepseer. Het is geen toeval dat veel programmeurs een grote hekel hebben aan marketing en pr, en ook niet dat grafisch ontwerpers vaak ontzet zijn door wat programmeurs er op hun eigen houtje van hebben gebakken.

En dat is jammer, want er zijn situaties waarin de vorm gelijk staat aan de inhoud, en de presentatie van een project is daar één van. Het eerste bijvoorbeeld dat een bezoeker te weten komt van een project is hoe de website eruitziet. Deze informatie wordt opgenomen voordat er ook maar enig inzicht is in de feitelijke inhoud van de site, dus voordat er ook maar iets van de tekst is gelezen of links zijn aangeklikt. Hoe onrechtvaardig dit ook kan lijken, mensen zullen nou eenmaal altijd een mening vormen op grond van de eerste indruk. De uitstraling van de website geeft aan of er zorg is besteed aan de presentatie van het project. Mensen hebben een extreem gevoelige antenne voor de hoeveelheid aandacht die is besteed. De meesten van ons kunnen in één oogopslag zeggen of een website snel in elkaar is geflanst of dat er serieus aandacht is besteed. Dit is het eerste stukje informatie dat je project publiceert en de indruk die dit maakt, is door associatie van invloed op de rest van het project.

Hoewel een groot deel van dit hoofdstuk gaat over de inhoud waarmee uw project van start zou moeten gaan, mag u dus niet vergeten dat het ook uitmaakt hoe het eruit ziet en aanvoelt. Omdat de website van het project bedoeld is voor twee verschillende soorten bezoekers, namelijk gebruikers en ontwikkelaars, moet er bijzondere aandacht worden besteed aan duidelijkheid en doelgroepgerichtheid. Hoewel dit niet de juiste tijd en plaats is voor een algemene verhandeling over webdesign, is één principe ervan de moeite van het vermelden waard, met name wanneer de site bestemd is voor meerdere doelgroepen (eventueel met overlap): bezoekers moeten min of meer kunnen zien waar een link naartoe gaat voordat ze erop klikken. Het moet bijvoorbeeld *alleen al door te kijken* naar de links naar gebruikersdocumentatie duidelijk zijn, dat ze naar gebruikersdocumentatie leiden en niet naar bijvoorbeeld ontwikkelaarsdocumentatie. Een project opzetten gaat voor een deel over het bieden van informatie, maar voor een deel ook over het bieden van gemak. Alleen de aanwezigheid al van bepaald standaardelementen op plaatsen waar deze worden verwacht, wekt vertrouwen bij gebruikers en ontwikkelaars. Uiteindelijk moeten zij beslissen of ze wel of niet aan het project willen deelnemen. Het laat zien dat het project zijn zaakjes goed heeft geregeld, dat het voorbereid is op de vragen die mensen gaan stellen en moeite heeft gedaan deze zo te beantwoorden dat er niet teveel inspanning van de vraagsteller wordt verwacht. Door uit te stralen dat het overal op voorbereid is, geeft het project een boodschap af: "Het is geen verspilling van je tijd wanneer je hieraan meedoet," en dat is precies wat mensen willen horen.

Maar eerst kijkt u rond

Voordat u met een open source-project van start gaat eerst een belangrijk waarschuwing:

Kijk altijd eerst heel goed rond of er al een project bestaat dat doet wat u wilt. Er bestaat een redelijke kans dat het probleem dat u nu wilt oplossen, iemand anders vóór u ook al heeft willen oplossen. Als ze het probleem hebben opgelost en de code uitgebracht hebben onder een vrije licentie, hoeft u het wiel niet opnieuw uit te vinden. Natuurlijk zijn er uitzonderingen: als u een project wilt opzetten als educatieve ervaring heeft u natuurlijk niets aan bestaande code. Ook kan het project dat u in gedachten hebt zo specialistisch zijn dat u zeker weet dat nog niemand dat heeft gedaan. Maar in het algemeen kan rondkijken echt geen kwaad en kunnen de voordelen enorm zijn. Wanneer de gebruikelijke zoekacties op internet niets opleveren, kunt u een kijkje nemen op http://freshmeat.net/ (een nieuwssite voor open source-projecten, waarover later meer), http://www.sourceforge.net/ en in de lijst van vrije software van de Free Software Foundation op http://directory.fsf.org/.
Zelfs als u niet precies kunt vinden wat u zoekt, kan het zijn dat u iets vindt dat zo dicht in de buurt komt dat het zinvoller is aan dat project deel te gaan nemen en daar functionaliteit aan toe te voegen dan helemaal bij nul te beginnen.

2.1 BEGINNEN MET WAT U HEBT

OK, u hebt rondgekeken, niets gevonden dat past bij wat u nodig hebt en besloten een nieuw project op te zetten.

Wat nu?

Het moeilijkste stuk van het lanceren van een open source-softwareproject is uw privévisie te transformeren naar een publieke visie. U of de organisatie waar u voor werkt, mag dan wel precies weten wat u wilt, het kan een hele opgave zijn om uw bedoelingen ook op een begrijpelijke manier aan de buitenwereld duidelijk te maken. Het is echter van cruciaal belang dat u hiervoor voldoende tijd uittrekt. U, en andere oprichters, moeten beslissen waar het project om draait, dat wil zeggen beslissen waar de grenzen liggen, wat het *niet* doet en wat wel, en dit in de vorm van een missieverklaring op papier zetten. Dit gedeelte is meestal niet echt moeilijk, hoewel er soms onuitgesproken vooronderstellingen en zelfs onenigheid aan het licht kunnen komen over de aard van het project, wat natuurlijk prima is: het is beter dit nu op te lossen dan later. De volgende stap is om het project zo te verpakken, dat het geschikt is voor consumptie. Dit is eerlijk gezegd een eentonig werkje.

Wat het zo bewerkelijk maakt is dat het voornamelijk bestaat uit het organiseren en documenteren van dingen die iedereen al lang weet; dat wil zeggen, iedereen die tot dat moment bij het project betrokken is geweest. Voor de mensen die het feitelijke werk doen, heeft dit dus eigenlijk geen direct voordeel. Zij hebben namelijk niet zoveel aan een README-bestand waarin een overzicht van het project wordt gegeven, noch aan een ontwerpersdocument of een gebruikershandleiding. Zij hebben geen behoefte aan een zorgvuldig opgezet codeschema waarmee wordt geconformeerd aan de informele maar wijdverspreide normen voor de distributie van softwarebroncode. De broncode is, ongeacht de opzet, voor hen sowieso in orde. Uiteindelijk zijn zij er al aan gewend. En als de code functioneert, weten ze ook hoe ze die moeten gebruiken. Het maakt hun zelfs niet uit als de fundamentele architecturale uitgangspunten van het project helemaal niet worden gedocumenteerd. Ook daaraan zijn ze al gewend.

Nieuwkomers aan de andere kant, hebben hier wel degelijk behoefte aan. Gelukkig hebben ze dit niet allemaal tegelijk nodig. Het is niet nodig om iedere mogelijke bron op papier te zetten voordat het project publiekelijk wordt gelanceerd. Alleen in een perfecte wereld gaat ieder nieuw open source-project van start met een gedetailleerd ontwerpdocument, een complete gebruikershandleiding (met speciale verwijzingen naar toepassingen die wel gepland maar nog niet geïmplementeerd zijn), prachtige en portbaar verpakte code die op ieder platform draait enz.

In de praktijk zou het aan elkaar knopen van al deze losse eindjes hinderlijk veel tijd kosten. Dit zijn dingen waarvan toch wel mag worden gehoopt dat vrijwilligers hiermee een handje zullen helpen zodra het project eenmaal loopt.

Wat echter *wel* nodig is, is dat er voldoende tijd en moeite wordt gestopt in de presentatie, waardoor nieuwkomers de eerste drempel van het onbekende over worden geholpen. Dit moet worden gezien als de eerste stap in het opstartproces, waarin het project de eerste minimale energiestoot krijgt toegediend. Ik heb deze eerste drempel ook wel *hacktivation energy* horen noemen: de hoeveelheid energie die een nieuwkomer erin moet steken voordat hij er iets voor terugkrijgt. Hoe lager de hacktivation energy van een project is, hoe beter. Uw eerste taak bestaat eruit deze hacktivation energy terug te brengen naar een niveau dat mensen aanspoort deel te nemen aan het project.

Iedere afzonderlijke subsectie hieronder beschrijft een belangrijk aspect van het opstarten van een nieuw project. Ze zijn in grote lijnen geordend in de volgorde waarmee een nieuwe bezoeker ermee wordt geconfronteerd, hoewel de volgorde waarin u ze uiteindelijk implementeert hiervan kan afwijken. U kunt de secties gebruiken als checklist. Wanneer u met een project van start gaat, ga dan gewoon de lijst langs en zorg ervoor dat ieder onderwerp aan bod komt, of dat u in ieder geval geen problemen hebt met de mogelijke gevolgen mocht u er een weglaten.

Een goede naam kiezen

Verplaats uzelf in iemand die net over uw project heeft gehoord, misschien doordat hij het toevallig tegen het lijf liep terwijl hij naar software zocht om een probleem op te lossen. Het eerste wat iemand dan ziet, is de naam van het project.

Een goede naam maakt uw project niet meteen tot een succes en een slechte naam zal echt niet de ondergang ervan betekenen (nou ja, een echt slechte naam zal dat misschien wel, maar we gaan er maar van uit dat niemand expres probeert een project te laten mislukken). Een slechte naam kan de acceptatie van het project echter vertragen, of omdat mensen hem niet serieus nemen, of gewoon omdat ze hem niet kunnen onthouden.

Een goede naam:

- geeft een idee van wat het project doet of heeft er in ieder geval een duidelijke relatie mee, zodat de naam hem later snel weer te binnen schiet;
- is makkelijk te onthouden. In dit geval kunt u niet om het feit heen dat Engels de standaardtaal is geworden van internet: ‘makkelijk te onthouden’ betekent ‘makkelijk te onthouden voor iemand die Engels kan lezen.’ Namen met woordspelingen die afhangen van de uitspraak van iemand met Engels als moedertaal kunnen abracadabra zijn voor mensen voor wie Engels niet de moedertaal is. Als de woordspeling bijzonder goed in het gehoor ligt en makkelijk te onthouden is, kan het wel de moeite waard zijn. Vergeet alleen niet dat veel mensen die de naam zien de woordspeling niet op dezelfde manier in hun hoofd horen als iemand met Engels als moedertaal;
- is niet gelijk aan namen van andere projecten en maakt geen inbreuk op handelsmerken. Dit is niet alleen fatsoenlijk maar ook juridisch verstandig. U wilt geen verwarring creëren over de identiteit. Het is vandaag de dag al moeilijk genoeg om bij te houden wat er allemaal op internet beschikbaar is zonder dat verschillende dingen dezelfde naam hebben.

  De bronnen die eerder zijn genoemd in het gedeelte ‘Maar eerst kijkt u rond’ kunnen goed van pas komen als u wilt uitzoeken of er al een project bestaat met de naam die u in gedachten hebt. U kunt gratis op handelsmerken zoeken op

http://www.nameprotect.org/ en http://www.uspto.gov/.
- is zo mogelijk beschikbaar als domeinnaam in de domeinen .com, .net en .org. U moet er één kiezen, waarschijnlijk .org, om mee te adverteren als de officiële website van het project. De andere twee moeten worden doorgelinkt naar deze site en zijn alleen bedoeld om te voorkomen dat anderen verwarring kunnen veroorzaken over de identiteit rond de naam van het project. Zelfs als u van plan bent de hosting van het project bij een andere site onder te brengen (zie het gedeelte 'Canned hosting') kunt u nog steeds specifieke domeinen registreren en deze doorlinken naar de hostingsite. Het helpt gebruikers als de URL makkelijk te onthouden is.

Zorg voor een duidelijke missieverklaring
Zodra mensen de website van het project hebben gevonden, gaan ze eerst op zoek naar een korte beschrijving, een missieverklaring, om (binnen 30 seconden) te kunnen beslissen of ze geïnteresseerd zijn. Deze moet op een prominente plaats op de eerste pagina vermeld staan, bij voorkeur direct onder de naam van het project.
De missieverklaring moet concreet, beperkend en vooral kort zijn. Hier een voorbeeld van een goede missieverklaring, van http://www.openoffice.org/:

We willen als een gemeenschap *hét internationale kantoorpakket ontwikkelen, dat beschikbaar is voor de belangrijkste besturingssystemen, met toegang tot alle functies en gegevens via open, componentgebaseerde API's en een op XML gebaseerd bestandsformaat.*

In slechts een paar woorden slaan ze de spijker op zijn kop, voornamelijk door in te spelen op de voorkennis van de lezer. Door 'als gemeenschap' te zeggen geven ze aan dat geen van de deelnemende partijen een dominante rol speelt in de ontwikkeling; 'internationaal' betekent dat de software mensen de kans geeft in meerdere talen en op verschillende plaatsen te werken en 'de belangrijkste besturingssystemen' houdt in dat het beschikbaar is voor Unix, Mac en Windows. De rest van de verklaring laat zien dat open interfaces en makkelijk te begrijpen bestandsindelingen een belangrijk onderdeel zijn van de doelstelling. Ze zeggen niet direct dat ze proberen een gratis alternatief te bieden voor Microsoft Office, maar de meeste mensen kunnen dit waarschijnlijk wel tussen de regels door lezen. Hoewel de missieverklaring op het eerste gezicht breed lijkt, worden de grenzen in feite duidelijk aangegeven: het woord '*kantoorpakket*' is heel concreet voor degene die bekend is met dergelijke software. Ook hier wordt de voorkennis van de lezer (in dit geval waarschijnlijk van MS Office) gebruikt om de missieverklaring beknopt te houden.

De aard van de missieverklaring hangt voor een deel af van de persoon die hem schrijft en niet alleen van de software die het beschrijft. Het is bijvoorbeeld logisch dat OpenOffice.org de woorden 'als gemeenschap' gebruikt, omdat het project werd gestart en nog steeds grotendeels wordt gesponsord door Sun Microsystems. Door deze woorden in de verklaring op te nemen geeft Sun aan dat het bedrijf zich bewust is van het feit dat mensen zich zorgen kunnen maken dat zij zullen proberen het proces te domineren. Met een dergelijke opmerking, die louter aangeeft dat men zich bewust is van een *potentieel* probleem, wordt het probleem al grotendeels ondervangen. Aan de andere kant hoeven projecten die niet worden gesponsord door één bedrijf dergelijke taal helemaal niet te gebruiken. Ontwikkeling door een gemeenschap is per slot van rekening de norm, dus normaal gesproken is er geen reden om dit in de missieverklaring te vermelden.

Vermeld dat het project vrij is
De mensen die nog steeds geïnteresseerd zijn nadat ze de missieverklaring hebben gelezen, zullen hierna meer informatie willen hebben, misschien gebruikers- of ontwerpersdocumentatie, en uiteindelijk iets willen downloaden. Maar voordat ze dat doen, willen ze zeker weten dat het om open source gaat.

De homepagina moet ondubbelzinnig duidelijk maken dat het om een open source-project gaat. Dit lijkt misschien voor de hand te liggen, maar u zult verbaasd zijn te zien hoeveel projecten dit vergeten. Ik heb open source-softwareprojecten gezien waarvan de homepagina niet alleen vergat te vermelden met welke afzonderlijke vrije licenties de software wordt gedistribueerd, maar zelfs dat de software vrij is. Soms werd deze informatie pas vermeld op de downloadpagina of de ontwikkelaarspagina, of een andere plaats die pas met een extra muisklik kon worden bereikt. In extreme gevallen werd er helemaal geen licentie vermeld op de website en was de enige manier om die te vinden de software te downloaden en er in te kijken.

Maak deze fout niet. Een dergelijke slordigheid kan u vele potentiële ontwikkelaars en gebruikers kosten. Vermeld direct aan het begin, direct onder de missieverklaring, dat het project gaat om 'vrije software' of 'open source software' en vermeld de exacte licentie. Voor een snelle handleiding over het kiezen van een licentie kunt u terecht in het gedeelte 'Een licentie kiezen en toepassen' verderop in dit hoofdstuk. Licentiekwesties worden uitgebreid behandeld in Hoofdstuk 9, *Licenties, auteursrechten en patenten.*

Op dit punt aangekomen heeft onze hypothetische bezoeker, waarschijnlijk in minder dan een minuut, besloten dat hij geïnteresseerd is om, laten we zeggen, nog eens een minuut of vijf te spenderen aan het project. In de volgende gedeelten wordt beschreven wat ze tegen zou moeten komen gedurende die vijf minuten.

Lijst met functies en vereisten
Er moet een korte lijst beschikbaar zijn met door de software ondersteunde functies (als iets nog niet klaar is, kunt u het wel in de lijst opnemen, maar met de vermelding '*gepland*' of '*in uitvoering*' ernaast) en de soort omgeving die vereist is om de software te kunnen gebruiken. De lijst met functies/vereisten kan worden gezien als de lijst die u iemand zou geven die vraagt om een korte samenvatting van de software. In de meeste gevallen is het een logische uitbreiding van de missieverklaring. De missieverklaring kan bijvoorbeeld de volgende zijn:

Een full-text indexer en zoekmachine creëren met een rich API voor gebruik door programmeurs door het bieden van zoekdiensten voor een groot aantal tekstbestanden.

De lijst met functies en vereisten bevat in dit geval de details, waarmee de draagwijdte van de missieverklaring wordt verduidelijkt:

Functies:

- Zoekt in platte tekst, HTML en XML
- Zoeken op woorden of zinnen
- (gepland) Fuzzy matching
- (gepland) Voortdurende updates van de indexen
- (gepland) Indexing van websites op afstand

Vereisten:

- Python 2.2 of hoger
- Voldoende schijfruimte voor de indexen (ongeveer 2x de omvang van de oorspronkelijke gegevens)

Met deze informatie kunnen lezers snel inschatten of de software voor hen zou kunnen werken en kunnen ze overwegen om ook als ontwikkelaar bij het project betrokken te zijn.

Ontwikkelingsstatus

Mensen willen altijd weten hoe ver een project gevorderd is. Voor nieuwe projecten willen ze weten hoe groot het gat is tussen de beloften van het project en de huidige realiteit. Voor gevorderde projecten willen ze weten hoe actief het wordt onderhouden, hoe vaak er nieuwe releases uitkomen, hoe snel er op bugmeldingen wordt gereageerd enz.

Om deze vragen te beantwoorden dient u te zorgen voor een pagina met de ontwikkelingsstatus, waarin de doelen en behoeften van het project voor de korte termijn worden weergegeven (het project kan bijvoorbeeld ontwikkelaars met een bepaalde expertise nodig hebben). Deze pagina kan ook een overzicht geven van de releases uit het verleden, met de functies, zodat bezoekers een indruk kunnen krijgen van hoe het project 'voortgang' definieert en hoe snel het daadwerkelijk vordert in verhouding met die definitie.

Wees niet bang om niet-klaar over te komen en geef niet toe aan de verleiding de ontwikkelingsstatus op te blazen. Iedereen weet dat software in fasen wordt ontwikkeld en niemand hoeft zich schamen om te zeggen: "Dit is alfasoftware met bekende bugs. Het loopt en werkt, soms, maar gebruik is voor eigen risico." Dergelijke taal schrikt het soort ontwikkelaar dat u in dit stadium nodig heeft niet af. En wat betreft de gebruikers, er is niet vervelender dan een project dat gebruikers lokt voordat de software klaar voor ze is. Als er eenmaal een reputatie van instabiliteit en bugs is gevestigd, komt u er niet zo makkelijk meer vanaf. Op de lange duur kunt u beter conservatief zijn. Het is altijd beter wanneer de software *stabieler* is dan de gebruiker verwacht dan *instabieler*, en prettige verrassingen zorgen voor de beste mond-tot-mondreclame.

Alfa en Bèta

De term *alfa* betekent over het algemeen de eerste release, waarmee gebruikers echt kunnen werken, en die over de bedoelde functionaliteit beschikt, maar waarvan ook bekend is dat hij bugs bevat. Het belangrijkste doel van alfasoftware is het genereren van feedback, zodat ontwikkelaars weten waar ze aan moeten werken. De volgende fase, *bèta*, betekent dat alle ernstige bugs in de software zijn hersteld, maar dat hij nog niet voldoende is getest voor de officiële release. Het doel van bètasoftware is dat dit de officiële release wordt wanneer er geen bugs meer worden gevonden, of dat er gedetailleerde feedback op komt voor de ontwikkelaars zodat ze de officiële release snel kunnen realiseren. Het verschil tussen alfa en bèta is vooral een kwestie van inzicht.

Downloads

De software moet gedownload kunnen worden als broncode in standaardbestandsindelingen. Wanneer het project zich nog in de opstartfase bevindt, zijn binaire (executable) pakketten niet nodig, behalve als de software zulke gecompliceerde kenmerken of afhankelijkheden heeft dat alleen het aan de gang krijgen al veel werk zou zijn voor de meeste mensen. (Als dat het geval is, zal het project sowieso moeite hebben om ontwikkelaars te vinden!)

Het distributiemechanisme moet zo gemakkelijk, standaard en goedkoop mogelijk zijn. Als u een ziekte wilt uitroeien, dan zult u er bij de distributie van het medicijn op letten dat het met een standaardinjectiespuit kan worden toegediend. Evenzo zou software moeten voldoen aan standaardbouw- en installatiemethodes. Hoe meer het van de normen afwijkt, des te groter de kans is dat potentiële gebruikers en ontwikkelaars het halverwege opgeven en afhaken.

Dit lijkt logisch, maar veel projecten doen tot aan het einde van het proces geen moeite hun installatieprocedures te standaardiseren, omdat ze zichzelf wijsmaken dat ze dit altijd later nog kunnen doen: *"Dat regelen we wel als de code bijna klaar is."* Zij realiseren zich niet dat door uitstel van het saaie werk aan de *build* en de installatieprocedures, het voltooien van de code feitelijk nog langer duurt, omdat ze ontwikkelaars afschrikken die anders misschien wel mee hadden willen werken aan de code. En wat nog verraderlijker is, is ze niet eens *weten* dat ze deze ontwikkelaar verliezen, omdat het proces uit een aaneenschakeling van non-events bestaat: iemand bezoekt de website, downloadt de software, probeert het te installeren, slaagt hier niet in, geeft het op en gaat weg. Maar wie anders dan de persoon zelf weet dat dit is gebeurd? Niemand van de mensen die aan het project werkt, realiseert zich dat iemands interesse en goede bedoelingen zomaar zijn verspild.

Saai werk met een hoog rendement moet altijd in een vroeg stadium worden gedaan. Het aanzienlijk verlagen van de toegangsdrempel van het project door het goed te verpakken geeft een zeer hoog rendement.

Wanneer u een downloadpakket uitbrengt, is het van extreem belang dat u de release een uniek versienummer toekent, zodat mensen altijd twee verschillende versies kunnen vergelijken en kunnen zien welke versie de laatste is. Een meer gedetailleerde uiteenzetting over versienummers vindt u in het gedeelte 'Releasenummers'. Meer informatie over het standaardiseren van build- en installatieprocedures vindt u in het gedeelte 'Downloadpakketten maken'. Beide onderwerpen zijn ook terug

te vinden in Hoofdstuk 7, *Downloadpakketten maken, releases en dagelijkse ontwikkeling.*

Toegang tot versiecontrole en bug tracker
Het downloaden van pakketten met de broncode is prima voor mensen die de software alleen willen installeren en gebruiken, maar het is niet gevoeg voor diegenen die hem willen debuggen of nieuwe functies willen toevoegen.

Screenshots van de broncode kunnen helpen, maar ze zijn nog steeds niet gedetailleerd genoeg voor een bloeiende ontwikkelaarsgemeenschap.

Mensen moet realtime toegang krijgen tot de laatste broncode. De manier om dat te doen is door gebruik te maken van een versiecontrolesysteem. De aanwezigheid van een anoniem toegankelijke versiegecontroleerde broncode is een signaal, zowel voor gebruikers als voor ontwikkelaars, dat dit project moeite doet de mensen te geven wat ze nodig hebben om te participeren. Als u niet meteen versiecontrole kunt aanbieden, plaats dan een melding dat u van plan bent dit snel te implementeren. De infrastructuur van versiecontrole wordt besproken in het gedeelte 'Versiecontrole' in Hoofdstuk 3, *Technische infrastructuur*.

Hetzelfde geldt voor de bug tracker van het project. Het belang van een bugopsporingssysteem zit 'm niet alleen in de bruikbaarheid voor de ontwikkelaars, maar ook in de boodschap die het geeft aan mensen die het project in de gaten houden. Voor veel mensen is een toegankelijke bugdatabase één van de sterkste signalen dat een project serieus moet worden genomen. Bovendien is het zo dat hoe groter het aantal bugs in de database is, des te beter het project eruitziet. Dit lijkt onlogisch, maar vergeet niet dat het aantal geregistreerde bugs van drie dingen afhangt: het absolute aantal aanwezige bugs in de software, het aantal gebruikers dat de software gebruikt en het gemak waarmee deze gebruikers nieuwe bugs kunnen melden. Van deze drie factoren spelen de laatste twee een grotere rol dan de eerste. Alle software van een bepaalde omvang en complexiteit heeft een in feite willekeurig aantal bugs, die er alleen nog maar op wachten te worden ontdekt. De feitelijk vraag is hoe goed een project in staat is deze bugs te registreren en er prioriteiten aan toe te kennen. Een project met een goed onderhouden bugdatabase (wat betekent dat er direct op bugs wordt gereageerd, dat dubbele bugs worden samengevoegd enz.) maakt daarom een betere indruk dan een project zonder bugdatabase of een bijna lege database.

Natuurlijk zal uw bugdatabase slechts een klein aantal bugs bevatten als uw project nog maar net is opgestart; daar is niet veel aan te doen. Als de statuspagina echter benadrukt dat het om een jong project gaat en als mensen naar de bugdatabase kijken en zien dat de meeste meldingen van recente datum zijn, kunnen zij daar uit afleiden dat het project beschikt over een in verhouding goed aantal registraties en zullen ze niet onnodig gealarmeerd worden door het lage absolute aantal geregistreerde bugs.

Merk op dat bugvinders vaak niet alleen worden gebruikt voor bugs in de software, maar ook voor verbeteringsverzoeken, wijzigingen in de documentatie, openstaande taken en nog veel meer. Meer informatie over het gebruik van een bug tracker vindt u in het gedeelte 'Bugvinder' in Hoofdstuk 3, *Technische infrastructuur*, daarom ga ik er hier niet op in. Het belangrijkste voor de presentatie van het project is het hebben van een bug tracker en ervoor te zorgen dat dit feit zichtbaar is op de homepagina van het project.

Communicatiekanalen
Bezoekers willen meestal weten hoe ze de mensen die bij het project betrokken zijn, kunnen bereiken. Verstrek de adressen van mailinglijsten, chatrooms en IRC-kanalen en alle andere mogelijke forums waar contact opgenomen kan worden met anderen die betrokken zijn bij de software. Maak duidelijk dat u en de andere auteurs van het project ook lid zijn van deze mailinglijsten, zodat mensen zien dat ze feedback kunnen geven die de ontwikkelaars ook bereikt. Uw aanwezigheid op deze lijst houdt niet in dat u zich verplicht alle vragen te beantwoorden of alle verzoeken om functies te implementeren. Uiteindelijk zullen de meeste gebruikers waarschijnlijk nooit lid worden van de forums, maar ze zullen gerustgesteld zijn door het feit dat het *kan* als het nodig is.

In de beginfase van het project is het niet nodig om afzonderlijke forums te hebben voor gebruikers en ontwikkelaars. Het is veel beter om iedereen die bij de software betrokken is ook met elkaar te laten praten, in één 'room'. Bij de vroege deelnemers is de grens tussen gebruiker en ontwikkelaar vaak niet zo duidelijk. Voor zover dit onderscheid wel kan worden gemaakt, is de ontwikkelaar-gebruikerverhouding in de beginfase van het project over het algemeen meer in het voordeel van eerstgenoemde dan later. Hoewel u er niet van uit kunt gaan dat iedere vroege deelnemer een programmeur is die wil meewerken aan het verbeteren van de software, kunt u er wel van uitgaan dat hij er in ieder geval in geïnteresseerd is om de ontwikkelingsdiscussies te volgen en een idee te krijgen van de richting van het project.

Omdat dit hoofdstuk alleen gaat over het opstarten van een project volstaat het te vermelden dat dergelijke communicatieforums dienen te bestaan. Later, in de sectie 'Omgaan met groei' in Hoofdstuk 6, *Communicatie*, gaan we uitzoeken waar en hoe u dergelijke forums kunt opzetten, de manier waarop ze gemodereerd of op een andere manier beheerd kunnen worden en hoe, wanneer de tijd daarvoor rijp is, het gebruikersforum moet worden afgesplitst van het ontwikkelaarsforum zonder een onoverbrugbare kloof te creëren.

Richtlijnen voor ontwikkelaars
Als iemand overweegt een bijdrage te leveren aan het project, zal hij op zoek gaan naar de richtlijnen voor ontwikkelaars. Richtlijnen voor ontwikkelaars zijn niet zozeer technisch als wel sociaal: ze geven aan hoe ontwikkelaars met elkaar en met de gebruikers omgaan en uiteindelijk hoe de dingen geregeld worden.

Dit onderwerp wordt meer gedetailleerd behandeld in het gedeelte 'Alles op papier zetten' in Hoofdstuk 4, *Sociale en politieke infrastructuur,* maar de belangrijkste elementen van de richtlijnen voor ontwikkelaars zijn:

• suggesties voor forums voor interactie met andere ontwikkelaars;

- instructies over *hoe* bugs moeten worden gerapporteerd en patches kunnen worden ingediend;
- een indicatie van hoe de ontwikkeling meestal plaatsvindt. Is het project een vriendelijke dictatuur, een democratie of nog iets anders.
- Met het woord 'dictatuur' wordt hier overigens niets negatiefs bedoeld. Er is helemaal niets mis met een dictatuur, waarbij één ontwikkelaar vetorecht heeft over alle wijzigingen. Veel succesvolle projecten werken op deze manier. Het belangrijkste is dat het project hier ook voor uitkomt. Een dictatuur die zich voordoet als democratie zal mensen tegen zich in het harnas jagen, terwijl een dictatuur die zegt een dictatuur te zijn prima kan functioneren, zolang de dictator competent is en vertrouwd wordt.
- Zie http://svn.collab.net/repos/svn/trunk/www/hacking.html voor een voorbeeld van bijzonder gedetailleerde ontwikkelaarsrichtlijnen, of http://www.openoffice.org/dev_docs/guidelines.html voor bredere richtlijnen, die zich meer richten op de supervisie en de geest van het meedoen en minder op technische zaken.
- Een programmeur een introductie geven over de software is een afzonderlijke kwestie, die wordt behandeld in het gedeelte 'Ontwikkelaarsdocumentatie' later in dit hoofdstuk.

Documentatie

Documentatie is van essentieel belang. Er moet altijd iets zijn dat mensen kunnen lezen, zelfs al is het elementair en onvolledig. Dit valt duidelijk onder het eerder genoemde 'eentonige' werk en is vaak waar een nieuw open source-project over struikelt. Een missieverklaring en een lijst met functies opstellen, een licentie kiezen, een overzicht geven van de ontwikkelingsstatus, dit zijn allemaal relatief kleine klusjes die geheel kunnen worden afgerond en waar meestal later niet meer op terug hoeft worden gekomen. Documentatie is echter nooit helemaal klaar, hetgeen een reden kan zijn waarom sommige mensen het maken ervan steeds maar weer uitstellen.

Het meest verraderlijke is dat de bruikbaarheid van de documentatie voor de schrijvers ervan vaak omgekeerd evenredig is aan de bruikbaarheid ervan voor de lezers. De meest belangrijke documentatie voor eerste gebruikers bestaat uit de beginselen: hoe de software snel kan worden geïnstalleerd, een overzicht van hoe hij werkt en misschien enkele instructies over het verrichten van veelvoorkomende taken. Dit is echter precies wat de *schrijvers* van de documentatie maar al te goed weten; zo goed, dat het moeilijk voor hen kan zijn om de dingen vanuit het perspectief van de lezer te bekijken. Zaken die voor de lezer uitvoerig beschreven zouden moeten worden, vinden zij vaak niet de moeite van het vermelden waard.

Er is helaas geen pasklare oplossing voor dit probleem. Iemand moet ervoor gaan zitten en alles opschrijven en vervolgens laten lezen door typisch nieuwe gebruikers om de kwaliteit ervan te testen. Gebruik een eenvoudig en makkelijk op te maken bestandsindeling zoals HTML, platte tekst, Texinfo of een variant van XML, wat soms de meest praktische oplossing is voor kleine, snelle en spontane verbeteringen. Dit is niet alleen goed om de drempels te weg te halen die de oorspronkelijke schrijvers ervan weerhouden periodieke verbeteringen aan te brengen, maar ook voor degenen die later bij het project komen en aan de documentatie willen werken.

Een manier om ervoor te zorgen dat de eerste basisdocumentatie wordt gemaakt, is om van tevoren de reikwijdte ervan te beperken. Daardoor lijkt het in ieder geval niet op een taak waaraan geen einde komt. Een goede vuistregel is dat deze minimaal moet voldoen aan de volgende criteria:

Vertel de lezer duidelijk hoeveel technische expertise van hem wordt verwacht.

Beschrijf helder en grondig hoe de software moet worden geïnstalleerd en vertel de gebruiker, ergens aan het begin van de documentatie, hoe een soort diagnostische test of enkelvoudig commando kan worden gedaan om te kijken of alles correct is geïnstalleerd. De documentatie over het opstarten van de software is in sommige gevallen nog belangrijker dan de feitelijke gebruikersdocumentatie. Hoe meer moeite iemand heeft moeten doen om de software te installeren en op te starten, des te vasthoudender hij zal zijn bij het uitzoeken van geavanceerde functionaliteit die niet goed is gedocumenteerd. Wanneer mensen afhaken, doen ze dat in de beginfase. Daarom hebben ze bij de eerste fasen, zoals de installatie, de meeste ondersteuning nodig.

Geef één voorbeeld in de vorm van een praktische oefening over hoe een veelvoorkomende taak kan worden uitgevoerd. Natuurlijk, veel oefeningen met veel taken zou nog beter zijn, maar als de tijd beperkt is, kiest u één taak die u grondig uitwerkt. Wanneer iemand gezien heeft dat de software gebruikt kan worden voor één ding, zullen ze op eigen houtje proberen uit te vinden wat ze nog meer kunnen en, als u geluk hebt, de documentatie zelf gaan schrijven. Wat ons bij het volgende punt brengt.

Geef de plaatsen aan waarvan bekend is dat de documentatie onvolledig is. Door de lezers te laten zien dat u zich bewust bent van de tekortkomingen laat u zien dat u de dingen vanuit hun perspectief bekijkt. Dit verzekert hen dat zij het project er niet van hoeven te overtuigen van wat belangrijk is en wat niet. Deze opmerkingen hoeven geen toezegging te zijn om de hiaten voor een bepaalde datum opgevuld te hebben, ze kunnen evengoed worden gezien als een open oproep voor hulp van vrijwilligers.

Het laatste punt is in feite van toepassing op het hele project en niet alleen op de documentatie. Het is in de wereld van open source normaal om de onvolkomenheden nauwkeurig bij te houden. U hoeft de tekortkomingen van het project niet te overdrijven, u moet ze alleen nauwgezet en feitelijk identificeren wanneer de context daarom vraagt (dat kan zijn in de documentatie, de bugtrackingdatabase of in een discussie via de mailinglijst). Niemand zal dit zien als probleem van het project, noch als toezegging om het probleem voor een bepaalde datum opgelost te hebben, behalve wanneer het project deze toezegging expliciet doet. Omdat iedereen die de software gebruikt de tekortkomingen zelf zal tegenkomen, is het beter dat zij daar psychologisch op voorbereid zijn. Daarmee maakt het project de indruk dat het precies weet waar het mee bezig is.

Bijhouden van veelgestelde vragen (FAQ)

Een *FAQ* (*Frequently Asked Questions*-document of veelgestelde vragen) kan één

van de meest waardevolle investeringen van een project zijn wat betreft educatief rendement. FAQ's zijn optimaal afgestemd op de feitelijke vragen van gebruikers en ontwikkelaars - dit in tegenstelling tot de vragen waarvan u verwacht dat ze worden gesteld. Daarom is een goed onderhouden FAQ vaak precies datgene waarnaar men op zoek is. De FAQ is vaak de eerste plaats waar gebruikers kijken wanneer ze een probleem tegenkomen, vaak zelfs nog voordat de officiële handleiding wordt geraadpleegd. Het is waarschijnlijk het document van uw project met de meest links vanuit andere websites.

Helaas kunt u geen FAQ opstellen aan het begin van het project. Goede FAQ's worden niet geschreven, ze groeien. Dit zijn per definitie reactieve documenten, die zich na verloop van tijd ontwikkelen in reactie op het praktische gebruik van de software. Omdat het onmogelijk is precies te voorspellen welke vragen mensen zullen stellen, is het ook onmogelijk een FAQ te schrijven vanaf nul.

Verspil daarom geen tijd door dit toch te proberen. Het kan echter wel handig zijn om een vrijwel geheel deel FAQ-sjabloon te maken, zodat mensen een duidelijk plek hebben waar ze vragen en antwoorden kunnen indienen nadat het project van start is gegaan. In deze fase is de belangrijkste eigenschap niet dat ze volledig zijn, maar dat ze bruikbaar zijn: als het toevoegen van FAQ's makkelijk is, zullen mensen het ook daadwerkelijk doen. (Goed onderhoud van FAQ's is een serieus en intrigerend probleem en wordt uitvoeriger behandeld in het gedeelte 'FAQ-manager' in Hoofdstuk 8, *Het managen van vrijwilligers.*)

Beschikbaarheid van documentatie

Documentatie moet op twee plaatsen beschikbaar zijn: online (direct vanaf de website) en in het downloadpakket van de software (zie het gedeelte 'Downloadpakketten maken' in Hoofdstuk 7, *Downloadpakketten maken, releases en dagelijkse ontwikkeling*). Documentatie moet online beschikbaar zijn in bladerbare vorm, omdat mensen documentatie vaak lezen *voordat* ze de software voor de eerste keer downloaden, om hen te helpen beslissen of ze de software sowieso willen downloaden. Zij moet echter ook bij de software zijn gevoegd, omdat bij het downloaden van het pakket in principe dat alles wat nodig is voor gebruik van het pakket aanwezig moet zijn (d.w.z. lokaal beschikbaar).

Voor online documentatie moet u ervoor zorgen dat er een link beschikbaar is die *alle* documentatie op één HTML-pagina laat zien (zet een opmerking als 'monolithisch', 'alles-in-één' of 'één grote pagina' bij de link, zodat mensen weten dat het even kan duren om de pagina te laden). Dit is van belang omdat mensen vaak naar een specifiek woord of zinsdeel willen zoeken in de hele documentatie. Over het algemeen weten mensen waar ze naar zoeken, ze zijn alleen vergeten in welke gedeelte het stond. Voor hen is het zeer frustrerend een HTML-pagina voor zich te krijgen met alleen de inhoud, vervolgens een andere pagina voor de inleiding en weer een andere pagina voor de installatie-instructies enz. Wanneer de pagina's zo zijn opgesplitst, is de zoekfunctie van de browser nutteloos. De indeling met afzonderlijke pagina's is handig voor mensen die al weten in welke gedeelte ze moeten zoeken, of die het gehele document van A tot Z willen doorlezen. Dit is echter *niet* de manier waarop de documentatie meestal wordt gebruikt. Het komt veel vaker voor dat iemand die vertrouwd is met de basisprincipes van de software terugkomt om te zoeken naar een specifiek woord of zinsdeel. Als u hun geen enkelvoudig en doorzoekbaar document biedt, maakt u het hun alleen maar moeilijker.

Documentatie voor ontwikkelaars

Documentatie voor ontwikkelaars wordt geschreven om programmeurs te helpen om de code te begrijpen, zodat ze deze kunnen repareren en uitbreiden. Dit is wat anders dan de ontwikkelaars*richtlijnen* die eerder aan bod zijn gekomen. Die hebben meer een sociaal dan een technisch karakter. Ontwikkelaarsrichtlijnen vertellen programmeurs hoe ze met elkaar moeten omgaan; ontwikkelaarsdocumentatie laat hun zien hoe ze met de code zelf om moeten gaan. Vaak worden de twee gemakshalve samengevoegd (zoals in het voorbeeld http://svn.collab.net/repos/svn/trunk/www/hacking.html dat eerder werd vermeld), maar het hoeft niet.

Hoewel ontwikkelaarsdocumentatie erg nuttig kan zijn, hoeft met de release niet te worden gewacht totdat deze klaar is. Zolang de oorspronkelijke auteurs beschikbaar (en bereid) zijn om vragen te beantwoorden over de code is dat voldoende om van start te gaan. Eerlijk gezegd is het feit dat men dezelfde vraag steeds weer moet beantwoorden de reden voor het schrijven van de documentatie. Maar ook voordat deze is geschreven, zullen mensen die vastbesloten zijn een bijdrage te leveren een manier vinden om de code te leren kennen. De drijfveer die ertoe leidt dat mensen de tijd nemen om een code te leren kennen, is het feit dat de code iets voor ze doet. Als mensen daar vertrouwen in hebben, zullen ze ook de tijd nemen dingen uit te zoeken. Als dit vertrouwen ontbreekt, zal zelfs de meest uitgebreide ontwikkelaarsdocumentatie ze niet overhalen om deel te nemen of te blijven.

Als u dus alleen tijd heeft om documentatie te schrijven voor één doelgroep, kies dan voor de gebruikers. Alle gebruikersdocumentatie is in feite ook ontwikkelaarsdocumentatie. Iedere programmeur die aan een deel van de software gaat werken, moet op de hoogte te zijn van het gebruik ervan. Wanneer programmeurs in een later stadium steeds dezelfde vragen stellen, kunt u de tijd nemen om enkele afzonderlijke documenten voor hen te schrijven.

Sommige projecten gebruiken 'wiki's' voor hun eerste documentatie, of zelfs als hun belangrijkste documentatie. In mijn ervaring werkt dit alleen als de wiki actief wordt geredigeerd door een aantal personen die het eens zijn over hoe de documentatie moet worden georganiseerd en welke 'toon' deze moet hebben. Raadpleeg voor meer informatie het gedeelte 'Wiki's' in Hoofdstuk 3, *Technische infrastructuur.*

Voorbeelden van output en screenshots

Als het project een grafische gebruikersinterface bevat of als het grafische of anderszins kenmerkende output genereert, plaats dan enkele voorbeelden hiervan op de website van het project. Voor een gebruikersinterface is dat een screenshot, voor de output kunnen dat screenshots of bestanden zijn. Beide spelen in op de behoefte van mensen aan directe bevrediging: een enkele screenshot kan overtuigender zijn dan eindeloze beschrijvende teksten en geklets op mailinglijsten, omdat een screenshot onbetwistbaar bewijs is dat de software echt werkt. Hij kan bugs bevatten, hij kan moeilijk te installeren zijn, de documentatie kan onvolledig zijn,

maar de screenshot bewijst dat als iemand genoeg moeite doet hij de software aan de gang kan krijgen.

Screenshots
Omdat het maken van screenshots een onmogelijke opgave kan lijken voordat u er werkelijk een paar hebt gemaakt, vindt u hier een paar instructies. Met behulp van Gimp (http://www.gimp.org/): ga naar File->Acquire->Screenshot, kies Single Window of Whole Screen en klik op OK. Uw eerstvolgende klik van de muis legt nu het venster of scherm vast waarop u klikt als afbeelding in Gimp. U kunt de afbeelding zo nodig bijsnijden of groter of kleiner maken, met behulp van de instructies op http://www.gimp.org/tutorials/Lite_Quickies/#crop.

Er zijn vele andere dingen die u op de website van het project kunt plaatsen als u daar de tijd voor heeft, of als dit om wat voor reden dan ook bijzonder nuttig blijkt: een nieuwspagina, een pagina met de projectgeschiedenis, een pagina met toepasselijke links, een zoekfunctie voor de website, een link voor donaties, etc. Geen van deze dingen is noodzakelijk tijdens de startfase, maar houdt u ze in gedachten voor de toekomst.

Canned Hosting
Er zijn enkele sites die vrije hosting en infrastructuur aanbieden voor open source-projecten: een weblocatie, versiecontrole, een bug tracker, een downloadlocatie, chatforums, regelmatige back-ups enz. De details variëren per site, maar de basisdiensten zijn bij allemaal gelijk. Door één van deze sites te gebruiken krijgt u veel dingen gratis. U geeft daar natuurlijk wel wat voor op, en dat is de fijnmazige controle over de gebruikerservaring. De hostingservice bepaalt welke software wordt gebruikt op de site en kan bepalen, of in ieder geval beïnvloeden, hoe de webpagina's van het project eruitzien en overkomen.
Raadpleeg de sectie 'Canned Hosting' in Hoofdstuk 3, *Technische infrastructuur* voor meer gedetailleerde informatie over de voor- en nadelen van canned hosting en een lijst met sites die dit aanbieden.

2.2 EEN LICENTIE KIEZEN EN TOEPASSEN

De bedoeling van dit gedeelte is om u een snelle, globale handleiding te geven voor het kiezen van een licentie. Lees Hoofdstuk 9, *Licenties, auteursrechten en patenten* voor de gedetailleerde wettelijke implicaties van de verschillende licenties en hoe de licentie die u kiest van invloed kan zijn op de mate waarin anderen uw software kunnen samenvoegen met andere vrije software.

Er bestaan vele, uitstekende licenties voor vrije software waaruit u kunt kiezen. De meeste daarvan hoeven hier niet aan bod te komen, omdat ze zijn opgesteld voor de specifieke juridische behoeften van een enkele organisatie of persoon en niet geschikt zijn voor uw project. We zullen ons hier beperken tot de meest gebruikte licenties. In de meeste gevallen zult u één van deze licenties willen gebruiken.

De 'Alles mag'-licenties
Als het voor u geen probleem is als de code van uw project wordt gebruikt in propriëtaire programma's kunt u een MIT/X-licentie gebruiken. Het is de eenvoudigste van verschillende minimale licenties, die niet veel meer doen dan het vastleggen van symbolische auteursrechten (zonder feitelijk het kopiëren te verbieden) en verklaren dat er geen garanties van toepassing zijn op de code. Raadpleeg voor meer informatie het gedeelte 'De MIT / X Window System-licentie'.

De GPL
Als u niet wilt dat uw code wordt gebruikt in propriëtaire programma's, gebruikt u de GNU General Public License (http://www.gnu.org/licenses/gpl.html). De GPL is waarschijnlijk de meest algemeen geaccepteerde licentie voor vrije software ter wereld. Dit is op zich al een groot voordeel, omdat veel potentiële gebruikers en mensen die eraan meewerken er reeds mee bekend zijn en daarom geen extra tijd hoeven te spenderen aan het lezen en begrijpen van uw licentie. Raadpleeg voor meer informatie het gedeelte 'De GNU General Public License' in Hoofdstuk 9, *Licenties, auteursrechten en patenten.*

Hoe u een licentie toepast op uw software
Zodra u een licentie hebt gekozen, moet u deze vermelden op de homepagina van uw project. U hoeft de feitelijke tekst van uw licentie hier niet te vermelden. Het is voldoende de naam van de licentie te noemen en een link op te nemen naar de volledige tekst van de licentie op een andere pagina.

Dit laat uw gebruikers weten onder welke licentie u van plan bent de software uit te brengen, maar het is niet voldoende voor wettelijke doeleinden. Daarvoor moet de software zelf voorzien zijn van de licentie. De standaardmanier om dit te doen is de volledige tekst van de licentie op te nemen in een bestand met de naam COPYING (of LICENSE) en een korte verwijzing op te nemen bovenaan ieder source-bestand, onder vermelding van de datum van het auteursrecht, de eigenaar en de licentie, en een vermelding van de plaats waar de volledige tekst van de licentie te vinden is.
Er zijn diverse manieren om dit te doen, we zullen hier slechts één voorbeeld bekijken. De GNU GPL vermeldt dat er bovenaan ieder source-bestand een mededeling moet worden opgenomen die er zo uit kan zien:

```
Copyright (C) <jaar>  <naam van de auteur>

Dit programma is vrije software. U kunt het distribueren en/of
wijzigen onder de voorwaarden van de GNU General Public License
zoals gepubliceerd door de Free Software Foundation, waarbij u
kunt kiezen uit versie 2 van de licentie of (naar eigen keus)
iedere latere versie.

Dit programma wordt gedistribueerd in de hoop dat het bruikbaar
is, maar ZONDER ENIGE VORM VAN GARANTIE, zelfs zonder de im-
pliciete garantie van VERKOOPBAARHEID of GESCHIKTHEID VOOR EEN
BEPAALD DOEL. Zie de GNU General Public License voor meer infor-
matie.
```

```
Samen met dit programma zou u een exemplaar ontvangen moeten
hebben van de GNU General Public License. Als dit niet het geval
is, schrijf dan naar Free Software Foundation, Inc., 59 Temple
Place, Suite 330, Boston, MA  02111-1307  USA
```

Er wordt niet specifiek vermeld dat het exemplaar van de licentie die u samen met het programma ontvangt het bestand COPYING is, maar over het algemeen wordt het daar ondergebracht. (U kunt de bovenstaande mededeling aanpassen en dit expliciet vermelden.) Dit voorbeeld geeft tevens een postadres waar een exemplaar van de licentie kan worden opgevraagd. Een andere veel gebruikte methode is om een link te geven naar een webpagina met de betreffende licentie. U kunt zelf het beste bepalen waar het meest permanente exemplaar van de licentie wordt bijgehouden en daar naar verwijzen. Dit zou een pagina op de website van uw project kunnen zijn. Over het algemeen hoeft de mededeling die u in ieder source-bestand opneemt niet exact gelijk te zijn aan de mededeling hierboven, zolang het begint met dezelfde vermelding van de houder en datum van de auteursrechten en de naam van de licentie en de plek waar de volledige licentie kan worden bekeken vermeld zijn.

2.3 DE TOON ZETTEN

Tot nu toe hebben we de eenmalige taken behandeld tijdens het opzetten van het project: het kiezen van een licentie, het maken van de website enz. De belangrijkste aspecten van het opstarten van een nieuw project zijn echter dynamisch. Het kiezen van een adres voor een mailinglijst is makkelijk. Ervoor zorgen dat de discussies op de lijst bij het onderwerp en productief blijven, is echter een hele andere zaak. Als het project voor buitenstaanders wordt geopend nadat het jaren binnenshuis is ontwikkeld, zullen de ontwikkelingsprocessen veranderen en zult u de bestaande ontwikkelaars moeten voorbereiden op die veranderingen.

De eerste stappen zijn de moeilijkste, omdat er nog geen gewoontes en verwachtingen zijn gecreëerd voor toekomstig gedrag. Stabiliteit binnen een project wordt niet bepaald door het formele beleid, maar door gedeelde, weinig tastbare inzichten die zich door de tijd ontwikkelen. Er zijn vaak ook schriftelijke regels, maar deze zijn vaak in wezen niet meer dan een aftreksel van de ongeschreven, zich steeds verder ontwikkelende onderlinge afspraken die sturing geven aan het project. Het schriftelijk beleid geeft geen definitie van het projectcultuur, maar eerder een beschrijving, en zelfs dan alleen nog bij benadering.

Er zijn een paar redenen waarom de dingen zo werken. Groei en een hoge omzet zijn niet zo schadelijk voor de ontwikkeling van sociale normen als wel gedacht wordt. Zo lang veranderingen niet te snel plaatsvinden, is er voldoende tijd voor instromers om te leren hoe alles in zijn werk gaat. Nadat ze dit hebben geleerd, zullen ze zelf ook helpen deze werkwijzen verder te versterken. Bedenk maar eens hoe kinderliedjes eeuwenlang hebben overleefd. Er zijn vandaag de dag kinderen die ongeveer dezelfde rijmpjes zingen als kinderen honderden jaren geleden, hoewel er nu geen kinderen in leven zijn die dat toen ook waren. Jongere kinderen horen de liedjes van oudere kinderen en zodra ze zelf ouder zijn, zullen zij ze op hun beurt ze weer voorzingen aan de andere jongere kinderen. Natuurlijk maken de kinderen geen deel uit van een bewust leerprogramma, maar de reden waarom liedjes de eeuwen overleven is dat ze regelmatig en herhaaldelijk worden overgedragen. De duur van een open source-softwareproject beslaat dan misschien geen eeuwen (dat weten we nu nog niet), maar de dynamiek van de overdracht is precies dezelfde. De verloopsnelheid is echter veel hoger en moet worden gecompenseerd door een meer actieve en doelbewuste overdracht.

Deze overdracht wordt ondersteund door het feit dat mensen, wanneer ze het project instappen, over het algemeen op zoek zijn naar sociale normen. Zo zitten mensen nou eenmaal in elkaar. Binnen iedere groep die bij elkaar wordt gehouden door een gemeenschappelijk inspanning gaan mensen instinctief op zoek naar gedragingen die hun kenmerken als onderdeel van die groep. Het doel van het vaststellen van deze gewoontes is ervoor te zorgen dat dit ‘groepsgedrag’ nuttig is voor het project. Zodra dit is vastgesteld houdt het zichzelf grotendeels in stand.

Hieronder volgen enkele voorbeelden van specifieke dingen die u kunt doen om goede gewoontes te creëren. De lijst is niet bedoeld als complete opsomming, maar als illustratie van het feit dat het creëren van een collectieve sfeer in de beginfase enorm nuttig kan zijn voor een project. Hoewel iedere ontwikkelaar feitelijk alleen in zijn kamertje aan het werk is, kunt u ze allemaal in belangrijke mate het gevoel geven dat ze allemaal samenwerken in dezelfde ruimte. Hoe meer ze dit gevoel krijgen, des te meer tijd ze aan het project zullen willen besteden. Ik heb deze specifieke voorbeelden gekozen omdat ik ze tegen ben gekomen in het Subversion-project (http://subversion.tigris.org/), waaraan ik heb deelgenomen en dat ik vanaf het allereerste begin heb geobserveerd. Ze zijn echter niet uniek voor Subversion. Situaties als deze komen voor in de meeste open source-projecten en moeten worden gezien als kansen om een project goed te beginnen.

Voorkom privédiscussies

Ook nadat u het project naar buiten hebt gebracht, zullen u en de andere oprichters er behoefte aan hebben moeilijke vragen onderling in besloten kring op te lossen. Dit is met name een feit in de eerste fase van het project, omdat er dan veel belangrijke beslissingen moeten worden genomen en er over het algemeen weinig vrijwilligers beschikbaar zijn die voldoende gekwalificeerd zijn om ze te beantwoorden. Alle overduidelijke nadelen van discussies via publieke lijsten doemen in alle mogelijke vormen voor u op: de vertraging die inherent is aan e-maildiscussies, de noodzaak voldoende tijd te geven aan het vormen van consensus, het gedoe met naïeve vrijwilligers die ten onrechte denken dat ze alles begrijpen (ieder project heeft zulke vrijwilligers, soms zijn ze de sterprogrammeurs van de toekomst, soms blijven ze voor altijd naïef), de persoon die niet kan begrijpen waarom u alleen probleem X wilt oplossen terwijl het overduidelijk deel uitmaakt van een groter probleem Y, en ga zo maar door. De verleiding om beslissingen achter gesloten deuren te nemen en ze later als voldongen feit te presenteren, of in ieder geval als resolute aanbeveling van een hechte en invloedrijke groep, zal in ieder geval zeer groot zijn.

Geef hier niet aan toe.

Hoe tijdrovend en zenuwslopend publieke discussies ook kunnen zijn, ze verdienen op de langere termijn bijna altijd de voorkeur. Belangrijke beslissingen in besloten kring nemen kan programmeurs bij uw project wegjagen. Geen enkele serieuze vrijwilliger blijft lang geïnteresseerd in een omgeving waar een geheime raad de belangrijke beslissingen neemt. Bovendien hebben openbare discussies positieve bijwerkingen die groter zijn dan het antwoord op de betreffende technische vraag:

- De discussie helpt nieuwe ontwikkelaars te trainen en op te leiden. U weet nooit hoeveel mensen meekijken bij de discussie. Zelfs als de meesten niet deelnemen, kunnen velen van hen het gesprek in stilte volgen en daarbij informatie verzamelen over de software.

- De discussie traint *u* in de kunst van het uitleggen van technische kwesties aan mensen die niet zo op de hoogte zijn van de software als u dat bent. Dit is een vaardigheid waarvoor veel oefening nodig is en u kunt dit niet oefenen door met mensen te praten die al weten wat u weet.

- De discussies en conclusies zullen altijd beschikbaar blijven in openbare archieven, waardoor voorkomen wordt dat toekomstige discussies in herhaling vallen. Zie het gedeelte 'Opvallend gebruik van archieven' in Hoofdstuk 6, *Communicatie*.

Ten slotte bestaat er natuurlijk ook nog de mogelijkheid dat iemand op de lijst een werkelijke bijdrage kan leveren aan de discussie door met een idee te komen dat u niet had verwacht. De kans hierop is moeilijk in te schatten. Het hangt af van de complexiteit van de code en de het vereiste specialisatieniveau van de deelnemers. Maar als anekdotisch bewijs acceptabel zou zijn, zou ik durven stellen dat de kans hierop groter is dan je intuïtief zou verwachten. In het Subversion-project waren wij (de oprichters) ervan overtuigd dat we te maken hadden met een aantal complexe problemen waarover we al maanden lang aan het tobben waren. Eerlijk gezegd betwijfelden we of er wel iemand op de nieuwe mailinglijst zou zijn die een daadwerkelijke bijdrage zou kunnen leveren aan de discussie. Daarom kozen we de weg van de minste weerstand en begonnen door middel van privé-e-mails te brainstormen over enkele technische kwesties, totdat een toeschouwer van het project[10] hier lucht van kreeg en vroeg of we de discussie wilden voeren via de publieke mailinglijst. We waren nogal sceptisch en daarom blij verrast door het aantal scherpzinnige opmerkingen en suggesties dat al vrij snel kwam. In veel gevallen hadden mensen ideeën waar wij niet eens op waren gekomen. Er bleek een aantal zeer slimme mensen op onze lijst te staan, die alleen maar zaten te wachten op een moment om toe te happen. Het klopt dat de daarop volgende discussies langer duurden dan wanneer we de discussie intern hadden gehouden, maar ze waren zoveel productiever dat het absoluut de moeite van de extra tijd waard was.

Zonder te vervallen in clichés als 'de groep is altijd slimmer dan het individu' (we hebben allemaal genoeg groepen gezien om wel beter te weten), moeten we wel erkennen dat er bepaalde activiteiten zijn waarin de groep beter is. Grootschalige controle door experts is hier één voorbeeld van, het snel genereren van een groot aantal ideeën een andere. De kwaliteit van de ideeën hangt natuurlijk af van de kwaliteit van het denkwerk dat eraan voorafging, maar u zult nooit weten welke soorten denkers u in huis heeft totdat u ze stimuleert met een uitdagend probleem.

Natuurlijk zijn er ook discussies die privé moeten blijven. In dit boek zullen we hiervan voorbeelden tegenkomen. Maar als uitgangspunt moet altijd gelden: *als er geen reden is om de discussie intern te voeren, moet hij publiekelijk worden gevoerd.*

Om dit te realiseren moet actie worden genomen. Het is niet voldoende om er alleen voor te zorgen dat uw eigen posts op de publieke lijst terechtkomen. U moet ook andermans onnodig privé gehouden onderonsjes een duwtje geven in de richting van de publieke lijst. Als iemand probeert een privédiscussie te beginnen, en er is geen reden de discussie privé te voeren, dan is het aan u om onmiddellijk een publieke discussie daarover te starten. U zou niet eens moeten reageren op het onderwerp voordat u erin bent geslaagd de discussie door te sluizen naar een publiek platform of hebt kunnen vaststellen dat het echt noodzakelijk is de discussie privé te voeren. Als u hier consequent in bent, zullen mensen snel doorhebben hoe het in zijn werk gaat en standaard de publieke forums gaan gebruiken.

Grofheden in de kiem smoren

Vanaf het allereerste begin van het publieke bestaan van uw project moet u een nultolerantiebeleid voeren ten aanzien van grof of beledigend gedrag op de forums. Nultolerantie betekent niet per se technische handhaving. U hoeft mensen niet van een mailinglijst te verwijderen wanneer ze andere deelnemers beledigen, of hun *commit access* af te nemen omdat ze kleinerende opmerkingen maken. (Theoretisch zou u uiteindelijk op dergelijke maatregelen moeten terugvallen, maar alleen nadat alle andere middelen hebben gefaald, waar aan het begin van het project per definitie geen sprake van kan zijn.) Nultolerantie betekent alleen dat incorrect gedrag niet onopgemerkt blijft. Wanneer iemand bijvoorbeeld een technisch bericht verstuurt waarin ook een *op de persoon gerichte* aanval is opgenomen tegen een andere ontwikkelaar van het project, is het noodzakelijk dat u de *persoonlijke* aanval *eerst* behandelt in uw reactie en pas daarna ingaat op de technische kwestie.

Helaas gebeurt het maar al te makkelijk en maar al te vaak dat constructieve discussies vervallen in destructieve ruzies. Mensen zeggen in e-mails dingen die ze iemand nooit recht in het gezicht zouden zeggen. Het onderwerp van de discussie versterkt dit effect alleen maar: voor technische kwesties geloven mensen vaak dat er slechts één correct antwoord is op de meeste vragen en dat onenigheid over dat antwoord alleen kan worden verklaard door onwetendheid of domheid. Het is maar een klein stapje tussen zeggen dat iemands idee stom is en zeggen dat de persoon zelf stom is. Het is in feite zelfs moeilijk te herkennen waar een technische discussie eindigt en de persoonlijke aanval begint. Dit is één van de redenen waarom drastische maatregelen of bestraffingen geen goed idee is. Wanneer u merkt dat zoiets gebeurt, plaatst u in plaats daarvan een post waarin u aangeeft hoe belangrijk het is de discussie vriendelijk te houden, zonder dat u iemand ervan beschuldigt opzettelijk gemeen te zijn. Dergelijke posts van de 'vriendelijke politie' hebben helaas de neiging te klinken als een kleuterjuf die een klas de les leest over goed gedrag:

laten we allereerst stoppen met (mogelijke) persoonlijke kritiek. Een voorbeeld hier-

van is zeggen dat J's ontwerp voor de Security Layer "naïef is en geen rekening houdt met de basisprincipes van computerbeveiliging." Dit kan wel of niet waar zijn, maar in beide gevallen is dit niet de manier om een discussie te voeren. J heeft zijn voorstel in goed vertrouwen gedaan. Als het tekortkomingen bevat, geef ze dan aan. Wij kunnen ze herstellen of een nieuw ontwerp maken. Ik weet zeker dat M J niet persoonlijk wilde beledigen, maar de woordkeuze was wat ongelukkig en we proberen de gesprekken hier opbouwend te houden.

En nu verder over het voorstel. Ik denk dat M gelijk had met ...

Hoe gekunsteld een dergelijke reactie ook kan klinken, het effect is opmerkelijk. Als u slecht gedrag consequent benoemt maar geen excuses of bevestiging verwacht van de beledigende partij, laat u mensen vrij om af te koelen en zich van hun betere kant te laten zien door zich de volgende keer fatsoenlijk te gedragen. En dat zullen ze dan ook doen. Eén van de geheimen om dit tot goed te doen, is ervoor te zorgen dat de discussie zelf niet het belangrijkste thema wordt. Het moet altijd een zijdelingse opmerking zijn, een korte inleiding tot het belangrijke deel van uw reactie. Merk terloops op "dat we discussies hier niet op deze manier voeren," maar stap dan over op het werkelijke onderwerp, zodat u mensen iets geeft waar het werkelijk over gaat en waar ze op kunnen reageren. Als iemand protesteert dat hij uw berisping niet heeft verdiend, weiger u dan mee te laten slepen in de woordenwisseling. U kunt óf helemaal niet reageren (als u denkt dat ze alleen stoom hoeven af te blazen en niet echt een antwoord nodig hebben), óf u zegt, voordat u overgaat tot het hoofdonderwerp, dat het u spijt dat u te sterk hebt gereageerd en dat het moeilijk is nuances te zien in e-mails. Dring nooit aan op een iemands bevestiging dat hij of zij zich incorrect heeft gedragen, publiekelijk noch privé. Als ze er zelf voor kiezen zich te verontschuldigen is dat geweldig, maar dit van ze eisen kan alleen maar tot wrok leiden.

Het algemene doel is om ervoor te zorgen dat goede omgangsvormen worden gezien als behorend bij de 'groepscultuur'. Dit is goed voor het project, omdat ontwikkelaars kunnen worden afgeschrikt (zelfs bij projecten waar ze graag aan zouden meewerken) door vuilbekkerij. Misschien weet u niet eens dat ze worden afgeschrikt. Iemand kan stilletjes lid zijn van de mailinglijst, tot de ontdekking komen dat je wel een erg dikke huid moeten hebben voor dit project en besluiten helemaal niet aan het project deel te willen nemen. De toon van de forums vriendelijk houden is een overlevingsstrategie voor de lange termijn, en veel makkelijker te realiseren zolang het project nog klein is. Zodra het onderdeel geworden is van de cultuur hoeft u niet de enige meer te zijn die dit propageert. Het zal door alle deelnemers in stand worden gehouden.

De code nakijken op verdachte onderdelen

Eén van de beste manieren om een productieve ontwikkelaarsgemeenschap te creëren is mensen naar elkaars code te laten kijken.

Er is enige technische infrastructuur voor nodig om dit doeltreffend te realiseren, in het bijzonder moet 'commit e-mails' zijn ingeschakeld. Raadpleeg voor meer informatie het gedeelte 'Commit e-mails'.

Het effect van commit e-mails is dat iedere keer wanneer iemand een wijziging doorvoert aan de broncode er een e-mail wordt verstuurd met het logbericht en de 'diff's' van de wijziging (zie diff in het gedeelte 'Woordenlijst versiecontrole'). Code review is de gewoonte om commit e-mails na te kijken op het moment dat ze binnenkomen, op zoek naar bugs of mogelijke verbeteringen.[11]

Code review heeft een aantal voordelen. Het is het meest voor de hand liggende voorbeeld van controle door experts in de wereld van open source en draagt direct bij aan de kwaliteit van de software. Iedere bug die binnensluipt in een onderdeel van de software is daar terechtgekomen doordat hij is gecommit en niet is ontdekt. Dus hoe meer mensen naar de doorgevoerde wijzigingen kijken des te minder bugs er binnen kunnen komen. Code review heeft echter ook een indirect voordeel: het laat mensen weten dat wat ze doen van belang is, omdat niemand de tijd zou nemen iets na te kijken tenzij die persoon belang heeft bij het effect. Mensen werken het beste wanneer ze weten dat anderen de tijd nemen om hun werk te evalueren. Reviews moeten publiek toegankelijk zijn. Zelfs in situaties waarbij ik in dezelfde ruimte zat als de andere ontwikkelaars en één van ons een commit had gemaakt, zorgden we ervoor om deze niet mondeling te evalueren maar in plaats daarvan de review naar de ontwikkelaarsmailinglijst te sturen. Iedereen heeft er baat bij dat de review daadwerkelijk zichtbaar is. Mensen volgen de commentaren en vinden daar soms onjuistheden in. Maar zelfs wanneer dat niet het geval is, herinnert het hun in ieder geval aan het feit dat de review een te verwachten standaardactiviteit is, zoals afwassen en grasmaaien.

In het Subversion-project was code review in het begin geen vaste gewoonte. Niemand kon garanderen dat iedere commit geëvalueerd zou worden, hoewel mensen soms wel wijzigingen bekeken als zij speciaal geïnteresseerd waren in dat deel van de code. Er slopen bugs in de software die echt opgespoord hadden kunnen en moeten worden. Een ontwikkelaar genaamd Greg Stein, die door voorgaande projecten bekend was met het belang van code review, besloot dat hij een voorbeeld zou stellen door *iedere regel van iedere commit* die in de code werd ingevoerd na te kijken. Op iedere commit, van wie dan ook, volgde al snel een e-mail naar de ontwikkelaarslijst van Greg, waarin de commit grondig werd ontleed en mogelijke problemen geanalyseerd werden. Van tijd tot tijd kreeg een bijzonder vernuftig stukje code een complimentje. Meteen vanaf het begin onderving hij bugs en niet-optimale code, die anders ongemerkt in de software zouden zijn geslopen. Opvallend genoeg klaagde hij nooit over het feit dat hij de enige was die iedere commit evalueerde, ook al koste het hem veel tijd. Maar hij liet niet na code review de hemel in te prijzen wanneer hij maar de kans kreeg. Al snel begonnen ook anderen, onder wie ikzelf, regelmatig commits te evalueren. Wat was onze motivatie daarvoor? Het was niet zo dat we wel moesten omdat Greg er voor zorgde dat wij ons zo schaamden. Maar hij had wel laten zien dat het evalueren van code een waardevolle tijdsbesteding was en dat iemand net zoveel aan het project kon bijdragen door andermans code te evalueren als door het zelf schrijven van nieuwe code. Toen hij dit eenmaal had aangetoond, werd dit de normale manier van werken, zelfs in die mate dat als er geen reactie kwam op een commit, de committer zich zorgen begon te maken en zelfs op de mailinglijst ging navragen of iemand al kans had gezien de commit te evalueren. Toen Greg later een andere baan kreeg, waardoor hij veel

minder tijd had voor Subversion, moest hij stoppen met de reviews. Tegen die tijd had de gewoonte echter zo wortel geschoten bij de rest van ons, dat het leek alsof we nooit iets anders hadden gedaan.

Begin vanaf de allereerste commit met reviews. Het soort problemen dat het makkelijkst op te sporen is door het evalueren van diff's zijn beveiligingskwetsbaarheden, *memory leaks*, ontoereikende uitleg of API-documentatie, off-by-one-fouten, caller-callee discipline mismatches en andere problemen waarvoor maar weinig context nodig is om ze op te kunnen sporen. Zelfs grootschaliger kwesties, zoals het feit dat verzuimd is herhaalde patronen samen te vatten naar één enkele locatie, zijn echter makkelijker op te sporen door iemand die regelmatig reviews doet, omdat diff's uit het verleden informatie geven voor huidige diff's.

Maakt u zich niet druk als u niets kunt vinden waar u commentaar op kunt leveren, of als u niet genoeg weet van ieder onderdeel van de code. Over iedere commit is over het algemeen wel iets te zeggen. Zelfs als u niets kunt vinden om vragen over te stellen, is er misschien wel iets waar u uw complimenten over wilt uitspreken. Het belangrijkste is dat u duidelijk maakt aan alle committers dat wat zij doen wordt gezien en begrepen. Natuurlijk ontslaat het doen van code reviews de programmeurs niet van de verantwoordelijkheid om hun eigen wijzigingen te evalueren en te testen voordat ze ze doorvoeren. Niemand mag het aan de code reviewers overlaten dingen te ontdekken die hij zelf had moeten ontdekken.

Wees bij het openen van een aanvankelijk gesloten project alert op de impact van de wijziging

Als u overgaat tot het openen van een bestaand project waarbinnen de ontwikkelaars eraan zijn gewend te werken in een gesloten source-omgeving, moet u ervoor zorgen dat iedereen begrijpt dat er een grote verandering aan staat te komen. Zorg er ook voor dat u begrijpt hoe zij zich voelen.

Probeert u zich voor te stellen hoe de situatie er voor hen uitziet. Voorheen werden alle beslissingen over de code en het ontwerp gemaakt binnen een groep van programmeurs die allemaal ongeveer evenveel van de software afwisten, die allemaal onder dezelfde druk stonden van het management en die op de hoogte waren van elkaars sterke en zwakke kanten. Nu vraagt u hen hun code bloot te stellen aan de kritische blik van willekeurige onbekenden, die hun mening alleen op de code baseren, zonder zich bewust te zijn van de kwesties binnen het bedrijf die misschien tot bepaalde beslissingen hebben geleid. Deze onbekenden zullen een heleboel vragen stellen; vragen die de oude groep ontwikkelaars heftig met de neus op de feiten drukken dat de documentatie waaraan ze zo hard hebben gewerkt nog steeds niet goed is (dit is onvermijdelijk). En alsof dat nog niet genoeg is, zijn de nieuwkomers ook nog eens onbekende, anonieme entiteiten. Als één van uw ontwikkelaars al onzeker was over zijn vaardigheden, stelt u zich dan eens voor hoe hij zich zal voelen wanneer nieuwkomers vallen over fouten in de code die hij heeft geschreven en, erger nog, dat ook nog in aanwezigheid van zijn collega's. Behalve wanneer u een team van perfecte programmeurs hebt, is dit onvermijdelijk. Sterker nog, dit is waarschijnlijk bij al uw programmeurs het geval. Dat is niet omdat ze slechte programmeurs zijn. De reden is gewoon dat ieder programma van een bepaalde omvang bugs bevat en een review door collega-programmeurs zal er daarvan altijd een aantal aan het licht brengen (zie het gedeelte 'De code nakijken op verdachte onderdelen' eerder in dit hoofdstuk). Tegelijkertijd zal er in het begin niet veel code van de nieuwkomers zelf worden geëvalueerd door de andere programmeurs, omdat ze pas zelf code kunnen gaan schrijven als ze wat meer vertrouwd zijn met het project. Op uw ontwikkelaars komt dit over alsof er alleen maar kritiek op hen afkomt en dat ze zelf geen kritiek kunnen leveren. Daardoor bestaat het gevaar dat de oude garde zich in gaat graven.

De beste manier om dit te voorkomen is om iedereen van tevoren te waarschuwen over wat er komen gaat, het uit te leggen, te vertellen dat het aanvankelijke ongemakkelijke gevoel volkomen normaal is, en te verzekeren dat het daarna beter zal gaan. Sommige van deze waarschuwingen zouden in een privégesprek gegeven moeten worden, voordat het project opengesteld wordt. Maar het kan ook nuttig zijn om mensen op de publieke mailinglijst eraan te herinneren dat dit een nieuwe manier van werken is voor het project, en dat het even kan duren voordat iedereen eraan is gewend. Het beste dat u kunt doen, is zelf het goede voorbeeld geven. Als u ziet dat uw ontwikkelaars niet genoeg vragen van nieuwkomers beantwoorden, helpt het nauwelijks als u ze zegt dat ze meer moeten antwoorden. Ze voelen misschien nog niet aan waarop ze wel en waarop ze niet moeten reageren. Het kan echter ook zijn dat ze nog niet goed hebben geleerd prioriteiten te stellen tussen het codeerwerk en de nieuwe verplichting tot externe communicatie. De beste manier om ze beter te laten participeren is door zelf te participeren. Wees actief op de publieke mailinglijst en zorg ervoor dat u ook enkele vragen beantwoordt. Wanneer u niet de expertise heeft om een vraag te beantwoorden, kunt u deze zichtbaar overdragen aan een ontwikkelaar die dat wel heeft. Let er daarbij op dat hij de vraag beantwoordt of in ieder geval reageert. Het is heel normaal dat de oude garde van ontwikkelaars geneigd is terug te vallen in privédiscussies; ze zijn niet anders gewend. Zorg ervoor dat u ook lid bent van eventuele interne mailinglijsten waar dit zou kunnen gebeuren, zodat u kunt vragen om dergelijke discussies naar de publieke lijst te verplaatsen.

Er zijn ook nog andere zaken die op de lange termijn een rol spelen bij het openen van voorheen gesloten projecten. Hoofdstuk 4, *Sociale en politieke infrastructuur* behandelt hoe betaalde en onbetaalde ontwikkelaars op succesvolle wijze kunnen samenwerken en Hoofdstuk 9, *Licenties, auteursrechten en patenten* bespreekt de noodzaak van juridische zorgvuldigheid bij het openen van een particuliere codebase die software kan bevatten die is geschreven door of 'eigendom' is van derden.

2.4 AANKONDIGEN

Zodra het project presentabel is — niet perfect, alleen persentabel — is het tijd om het aan de buitenwereld bekend te maken. In feite is dit een heel eenvoudig proces: ga naar http://freshmeat.net/, klik op Submit in de bovenste navigatiebalk en vul een formulier in waarin u uw nieuwe project bekendmaakt. Freshmeat is de plaats waar iedereen kijkt voor aankondigingen van nieuwe projecten. U hoeft slechts de aandacht te trekken van een paar mensen en uw project wordt via mond-tot-mondreclame bekend.

Als u op de hoogte bent van mailinglijsten of nieuwsgroepen waar een aankondiging van uw project binnen de discussie en het interessegebied valt, kunt u daar een post plaatsen. Zorg er echter voor om niet meer dan één post per forum te plaatsen en de mensen naar uw eigen forums door te sturen voor verdere discussie (door het reply-to-veld in te stellen). De post moet kort en to the point zijn:

```
To: discuss@lists.example.org
Subject: [ANN] Scanley full-text indexer project
Reply-to: dev@scanley.org

Dit is een eenmalige post om de lancering aan te kondigen van
het Stanley-project, een open source full-text indexer en zoek-
machine met een 'rich API', voor gebruik door programmeurs, die
een zoekfunctie biedt voor een groot aantal tekstbestanden.
Scanley is nu een werkende code, wordt actief ontwikkeld en is
op zoek naar zowel ontwikkelaars als testers.

Website: http://www.scanley.org/

Functies:
   - Zoekt in platte tekst, HTML en XML
   - Zoeken op woorden of zinnen
   - (gepland) Fuzzy matching
   - (gepland) Voortdurende updates van de indexen
   - (gepland) Indexing van websites op afstand

Vereisten:
   - Python 2.2 of hoger
   - Voldoende schijfruimte voor de indexen (ongeveer 2x de om-
     vang van de oorspronkelijke gegevens)

Voor meer informatie brengt u een bezoek aan scanley.org.

Hartelijk dank,
-J. Random
```

(Raadpleeg het gedeelte 'Publiciteit' in Hoofdstuk 6, *Communicatie voor advies over de aankondiging van volgende releases en andere gebeurtenissen*.)
Er is in de wereld van de vrije software veel discussie gaande over of een project pas gepubliceerd mag worden wanneer er werkende code beschikbaar is, of dat het handig kan zijn een project al te openen tijdens de ontwerp- en discussiefase. Ik heb altijd gedacht dat het belangrijk was om te beginnen met werkende code, dat dat het onderscheid maakte tussen een succesvol project en speeltje, en dat serieuze ontwikkelaars zich alleen in zouden willen laten met software die al iets concreets zou kunnen.

Dit bleek niet het geval te zijn. Bij het Subversion-project begonnen we met een ontwerpdocument, een clubje geïnteresseerde en goed communicerende ontwikkelaars, een heleboel poeha en geen enkele werkende code. Tot mijn grote verrassing trok het project vanaf het eerste begin actieve deelnemers aan en op het moment dat we beschikten over werkende code waren er al heel wat vrijwillige ontwikkelaars nauw betrokken bij het project. Subversion is niet het enige voorbeeld. Het Mozilla-project werd ook zonder werkende code gelanceerd en is nu een succesvolle en populaire webbrowser.

Door dit soort voorbeelden moet ik dus afstand nemen van de aanname dat werkende code een absolute noodzaak is voor het lanceren van een project. Werkende code is nog steeds één van de beste uitgangspunten voor succes en het is een goede vuistregel om te wachten met het aankondigen van het project totdat u werkende code hebt. Er zijn echter omstandigheden waarin het lanceren van het project in een eerder stadium verstandig is. Ik denk wel dat minimaal een goed ontwikkeld ontwerpdocument, of anders één of andere vorm van een codekader noodzakelijk is. Natuurlijk kan dit worden gewijzigd aan de hand van publieke reacties, maar er moet wel iets concreets zijn, iets dat tastbaarder is dan goede bedoelingen, iets waar mensen hun tanden in kunnen zetten.

Wanneer u een project aankondigt, verwacht dan niet meteen dat er hordes vrijwilligers op af komen. Meestal is het gevolg van een aankondiging dat u een paar sporadische verzoeken om informatie krijgt en dat wat meer mensen lid worden van uw mailinglijst, maar verder blijft alles ongeveer bij het oude. Na verloop van tijd zult u echter merken dat het aantal programmeurs en gebruikers geleidelijk toeneemt. De bekendmaking is meer het planten van een zaadje. Het kan lang duren voordat het nieuws bekend wordt. Als het project de betrokkenen consequent beloont, zal het nieuws uiteindelijk steeds bekender worden. Mensen willen het namelijk graag aan anderen laten weten als ze iets goeds hebben gevonden. Als alles goed gaat zal de dynamiek van de exponentiële communicatienetwerken uw project langzaam transformeren in een complexe gemeenschap, waar u niet per se iedereen kent en niet langer iedere conversatie kunt bijhouden. De volgende hoofdstukken gaan over het werken in een dergelijke omgeving.

10| We zijn nog niet aangekomen bij het gedeelte over het bedanken van mensen, maar hier vast een oefening over wat ik later uit zal leggen: de naam van degene die dit heeft opgemerkt is Brian Behlendorf. Hij was ook degene die het algemene belang aangaf van het publiek houden van alle discussies tenzij er een bijzondere reden voor privacy is.

11| Hoe dan ook, dit is hoe code review normaal gesproken wordt gedaan in open source-projecten. In meer gecentraliseerde projecten kan 'code review' ook betekenen dat meerdere mensen bij elkaar komen en prints van de broncode bestuderen, op zoek naar specifieke problemen of patronen.