4

SOCIALE EN POLITIEKE INFRASTRUCTUUR

De eerste vragen die mensen meestal stellen over open source software zijn: “Hoe werkt het? Wat houdt het project gaande? Wie neemt de beslissingen?”. Ik voel met niet zo prettig bij nietszeggende antwoorden over meritocratie, de geest van samenwerking, de code die voor zichzelf spreekt enz. Feit is nou eenmaal dat deze vraag moeilijk te beantwoorden is. Meritocratie, samenwerken en werkende code maken er allemaal deel van uit. Toch zeggen ze weinig over hoe een project van dag tot dag werkelijk functioneert en ze zeggen helemaal niets over hoe conflicten worden opgelost.

In dit hoofdstuk probeer ik de onderliggende principes te laten zien die alle succesvolle projecten over het algemeen hebben. Ik bedoel succesvol niet alleen wat betreft de technische kwaliteit, maar ook ten aanzien van de operationele gezondheid en het overlevingsvermogen. De operationele gezondheid is het voortdurende vermogen van het project om nieuwe codebijdragen in te passen, nieuwe ontwikkelaars aan te trekken en te reageren op binnenkomende bugrapporten. Het overlevingsvermogen is het vermogen van het project om onafhankelijk van een individuele participant of sponsor te blijven bestaan. Anders gezegd, het is de kans dat het project blijft voortbestaan ook als alle leden die het project gestart zijn andere bezigheden vinden. Technisch succes is niet moeilijk te bewerkstelligen, maar zonder een solide basis van ontwikkelaars en een sociaal fundament kan het gebeuren dat een project de groei die het gevolg is van het aanvankelijke succes of het vertrek van charismatische figuren niet aankan.

Er zijn verscheidene manieren om dit soort succes te bewerkstelligen. Voor sommige daarvan is een formele bestuurlijke structuur nodig, waarbij discussies worden opgelost, nieuwe ontwikkelaars uitgenodigd (soms om te vertrekken) en nieuwe functies gepland enz. Voor andere is een minder formele structuur nodig en meer bewuste terughoudendheid om een atmosfeer van eerlijkheid te creëren waarop mensen kunnen vertrouwen. Beide manieren leiden tot hetzelfde resultaat: een besef van institutionele duurzaamheid, ondersteund door gewoontes en procedures die door alle deelnemers goed begrepen worden. Deze eigenschappen zijn zelfs nog belangrijker in zelforganiserende systemen dan in centraal gecontroleerde, omdat in een zelforganiserend systeem iedereen zich bewust is van het feit - in ieder

geval gedurende een bepaalde tijd - dat een paar rotte appels de hele mand kunnen bederven.

Afsplitsbaarheid of forkability
Het onmisbare ingrediënt dat ontwikkelaars in een open source-softwareproject samenbindt en ze bereidwillig maakt om compromissen te sluiten als dat nodig is, is de *forkability* van de code: dit is de mogelijkheid dat iedereen een kopie van de broncode kan maken en daarmee een concurrerend project *(fork)* kan beginnen. Het tegenstrijdige is dat bij een open source-softwareproject de *mogelijkheid* van een fork belangrijker is dan daadwerkelijke forks, die zeer zeldzaam zijn. Omdat een fork slecht is voor iedereen (de redenen hiervoor worden in detail behandeld in het gedeelte 'Forks' in Hoofdstuk 8*, Het managen van vrijwilligers*) zullen naarmate de dreiging van een fork toeneemt steeds meer mensen bereid zijn om compromissen te sluiten om die te voorkomen.

Forks, of beter gezegd de mogelijkheid voor het ontstaan van een fork, zijn de reden dat er bij een open source-softwareproject geen echte dictator kan bestaan. Dat klinkt misschien verrassend, met name gezien het feit dat het vrij gebruikelijk is dat iemand in een open source-project de 'dictator' of de 'tiran' genoemd wordt. Maar dit soort tirannie is anders en verschilt nogal van de conventionele betekenis van het woord. Stelt u zich eens voor: de onderdanen van een koning kunnen op ieder willekeurig moment een kopie maken van diens hele koninkrijk, daar naartoe verhuizen en naar believen zelf regeren. Zou zo'n koning niet heel anders regeren dan een collega wiens onderdanen gedwongen waren onder zijn bewind te blijven, ongeacht wat hij doet?

Daarom zijn zelfs projecten die formeel niet zijn georganiseerd als een democratie, in de praktijk toch een democratie als het op het nemen van belangrijke beslissingen aankomt. Vermenigvuldigbaarheid impliceert forkability, forkability impliceert consensus. Het kan zijn dat iedereen bereid is zich op één leider te verlaten (het bekendste voorbeeld daarvan is Linus Torvalds in Linux kernel development), maar dit is omdat men daarvoor *gekozen* heeft, op een volkomen niet-cynische, niet-kwaadaardige manier. De dictator heeft geen magische greep op het project. Een belangrijke eigenschap van alle open source-licenties is dat ze de ene partij niet meer zeggenschap geven over hoe de code veranderd of gebruikt mag worden dan de andere. Als de dictator plotseling slechte beslissingen gaat nemen, heeft dit rusteloosheid tot gevolg, gevolgd door opstand en een fork. Uitzonderingen daargelaten, lopen de zaken zelden zo hoog op, omdat de dictator nagenoeg altijd eerst een compromis zoekt.

Maar alleen omdat forkability de zeggenschap beperkt die één persoon kan uitoefenen, wil dat nog niet zeggen dat er geen belangrijke verschillen zijn in hoe projecten bestuurd worden. U wilt niet bij iedere beslissing met de ultieme vraag geconfronteerd worden wie er een fork overweegt. Dat zou erg vermoeiend worden en alleen maar de aandacht afleiden van het echte werk. De volgende twee gedeeltes gaan over verschillende manieren om projecten op zo'n manier te organiseren dat het nemen van de meeste beslissingen soepel verloopt. Het gaat om twee enigszins geïdealiseerde, extreme voorbeelden. Veel projecten zitten hier ergens tussenin.

4.1 VRIENDELIJKE DICTATORS

Het *vriendelijkedictatormodel* is precies wat het lijkt. De uiteindelijke beslissingsbevoegdheid ligt bij één persoon, van wie op basis van zijn persoonlijkheid en ervaring verwacht wordt dat hij deze verstandig gebruikt.

Alhoewel 'vriendelijke dictator' (VD) de standaardterm is voor deze functie, is het beter te denken in termen van een 'door de gemeenschap goedgekeurde scheidsrechter' of 'rechter'. Over het algemeen nemen VD's niet alle beslissingen; vaak nemen ze zelfs niet eens de meeste beslissingen. Het is onwaarschijnlijk dat één persoon over voldoende kennis beschikt om constant de juiste beslissingen te nemen ten aanzien van alle aspecten van het project. Wat zeker is, is dat goede ontwikkelaars niet bij een project betrokken blijven, tenzij ze invloed kunnen uitoefenen op de koers ervan. Daarom valt er voor een VD over het algemeen weinig te dicteren. In plaats daarvan laat hij, wanneer mogelijk, de problemen zichzelf oplossen door middel van discussies en experimenten. Aan deze discussies neemt hij zelf deel, maar dan als gewone ontwikkelaar. Vaak voegt hij zich naar een ander die meer kennis heeft van dat specifieke gebied. Alleen wanneer het duidelijk wordt dat er geen consensus bereikt gaat worden en dat de meerderheid *wil* dat iemand de beslissing neemt ter wille van de voortgang van de ontwikkeling, dan houdt hij zijn poot stijf en zegt 'zo gaan we het doen'. Terughoudendheid in het nemen van beslissingen op deze manier is een kenmerk van vrijwel alle succesvolle VD's en het is één van de redenen waarom ze hun positie kunnen behouden.

Wie is geschikt als vriendelijke dictator?
VD zijn vraagt om een combinatie van eigenschappen. In de eerste plaats heeft men een goed ontwikkeld gevoel nodig voor de eigen invloed op het project, wat op zijn beurt weer leidt tot zelfbedwang. In het begin van een discussie moet de VD vermijden om met grote stelligheid meningen en conclusies naar voren te brengen. Anders krijgen anderen het gevoel dat het zinloos is om een afwijkende mening te uiten. Mensen moeten zich vrij voelen om ideeën te kunnen ventileren, ook stomme ideeën. Het is onvermijdelijk dat de VD zelf van tijd tot tijd ook met een dom idee op de proppen komt. Daarom moet hij tevens over het vermogen beschikken om een eigen fout te herkennen en te erkennen (*iedere* goede ontwikkelaar zou die eigenschap natuurlijk moeten hebben, vooral als hij lang bij het project betrokken blijft). Het verschil is dat de VD zich af en toe een foutje kan permitteren zonder zich zorgen te hoeven maken zijn geloofwaardigheid op lange termijn. Ontwikkelaars met minder ervaring voelen zich misschien minder zeker. Daarom moet de VD zijn kritiek of die beslissingen die negatief uitpakken voor de minderervaren ontwikkelaar goed wegen, zowel op technisch als op psychologisch vlak.

De VD hoeft *niet* degene te zijn met de beste technische vaardigheden binnen het project. Hij moet bekwaam genoeg zijn om aan de code zelf te kunnen werken en een verandering te kunnen begrijpen, te bekritiseren en af te wegen, maar dat is alles. De VD-functie wordt niet verkregen of behouden op basis van intimiderend goede programmeervaardigheden. Wat *wel* belangrijk is, is ervaring en een zesde zintuig voor ontwerp als zodanig, niet noodzakelijkerwijs de vaardigheid om een goed ontwerp te produceren, maar de vaardigheid om het te herkennen, ongeacht waar het vandaan komt.

Het is meestal zo dat de VD ook de oprichter van het project is, maar dit is meer een logisch dan een causaal verband. De verschillende kwaliteiten die iemand in staat stellen om met succes een project te beginnen (technische deskundigheid, het vermogen andere mensen over te halen zich bij het project aan te sluiten enz.) zijn precies de kwaliteiten die een VD nodig heeft. En oprichters van een project hebben natuurlijk automatisch al het voordeel van anciënniteit, waardoor voor alle betrokkenen VD-schap van deze persoon vaak de weg van de minste weerstand is.

Vergeet niet dat wat betreft een fork het mes aan twee kanten snijdt. Een VD kan net zo makkelijk een fork van een project maken als iemand anders en sommige VD's hebben dat dan ook gedaan, toen ze het gevoel kregen de richting die zij met het project op wilden niet strookte met die van de meerderheid van de ontwikkelaars. Omdat forkability bestaat, maakt het niet uit of de VD systeembeheerdersprivileges van de belangrijkste servers van het project of niet. Mensen denken vaak dat controle over de server de ultieme macht binnen een project betekent, maar in feite is het niet relevant. De mogelijkheid om op een bepaalde server wachtwoorden van mensen toe te voegen of te verwijderen, heeft alleen betrekking op de kopie die zich op die server bevindt. Voortdurend misbruik van die macht, of dat nu door de VD is of door iemand anders, zou alleen maar leiden tot verplaatsing van de ontwikkeling naar een andere server.

Of uw project een VD zou moeten hebben of misschien beter geleid zou worden met een minder gecentraliseerd systeem, hangt voornamelijk af van wie beschikbaar is voor de functie. Een algemene regel is dat als het voor iedereen overduidelijk is wie de VD zou moeten zijn, dat de weg is die u in moet slaan. Maar als er geen overduidelijke kandidaat is voor het VD-schap, kan het project beter gebruik maken van een gedecentraliseerd beslissingsproces zoals wordt beschreven in het volgende gedeelte.

4.2 CONSENSUSDEMOCRATIE

Wanneer projecten een tijdje lopen, hebben ze de neiging het model van VD-schap te willen verlaten ten gunste van een open, democratisch systeem. Dit is niet noodzakelijkerwijs uit ontevredenheid over een bepaalde VD. Een op de groep gebaseerd bestuur is nou eenmaal 'evolutionair stabieler', om maar eens een metafoor uit de biologie te gebruiken. Iedere keer dat een VD aftreedt of probeert de verantwoordelijkheid voor het nemen van beslissingen gelijkmatiger te verdelen, is dat een kans voor de groep om een nieuw, niet-dictatoriaal systeem te vestigen: om een nieuwe grondwet vast te stellen als het ware.

De groep zal misschien niet van de eerste gelegenheid gebruik maken, of van de tweede, maar uiteindelijk gebeurt het toch. En als dat eenmaal het geval is, is er meestal geen weg terug meer.

Gezond verstand zegt waarom: als een groep van N mensen een persoon speciale macht toekent, betekent dat dat N - 1 mensen allemaal instemden met het feit dat hun individuele invloed minder zou worden. Mensen doen dat over het algemeen liever niet. En zelfs als ze het doen, dan nog is het resulterende dictatorschap slechts voorwaardelijk. De groep heeft de dictator gezalfd en het is duidelijk dat de groep de dictator ook kan afzetten. Wanneer het project eenmaal overgestapt is van een charismatisch leider naar een meer formeel, op de groep gebaseerd leiderschap, keert men maar zelden op zijn schreden terug.

Hoewel deze systemen in detail zeer verschillend functioneren, hebben ze twee elementen gemeen. Ten eerste werkt de groep meestal met consensus; ten tweede kan men terugvallen op een formele stemprocedure mocht geen consensus bereikt kunnen worden.

Consensus betekent alleen maar een overeenkomst waarmee iedereen bereid is te leven. Een groep heeft consensus bereikt over een bepaalde kwestie wanneer iemand voorstelt dat consensus is bereikt en niemand dit tegenspreekt. De persoon die consensus voorstelt moet uiteraard helder verwoorden wat deze consensus inhoudt en welke acties er ondernomen moeten worden, voor zover dit niet reeds duidelijk is.

De meeste discussies in een project gaan over technische onderwerpen, zoals de juiste manier om een bepaalde 'bug' te herstellen, het wel of niet toevoegen van een nieuwe functie, hoe strikt interfaces gedocumenteerd moeten worden enz. Op consensus gebaseerd bestuur werkt goed omdat het naadloos aansluit op de technische discussie zelf. Aan het eind van de discussie is er vaak eenstemmigheid over de te volgen koers. Meestal zal iemand een concluderende post voor de mailinglist maken. Dit is tegelijkertijd een samenvatting is van wat er beslist is en impliciet een voorstel tot consensus. Dit is tevens de laatste kans om te zeggen 'Wacht even, ik ben het daar niet mee eens. We moeten hier nog eens goed naar kijken.'

Voor kleine en niet-controversiële beslissingen is het voorstel tot consensus impliciet. Wanneer een ontwikkelaar bijvoorbeeld spontaan een bugfix commit, dan is deze commit op zichzelf al een voorstel tot consensus: "Ik neem aan dat we het er allemaal over eens zijn dat deze bug hersteld moet worden en dat dit de manier is om hem te herstellen." Natuurlijk zegt de ontwikkelaar dit niet daadwerkelijk. Hij repareert de bug gewoon. De anderen binnen het project hoeven niet expliciet akkoord te gaan: wie zwijgt, stemt toe. Als iemand een verandering commit waarover achteraf *geen* consensus blijkt te bestaan, dan is de oplossing simpelweg, dat deze verandering binnen het project opnieuw ter discussie wordt gesteld alsof de aanpassing nog niet was gecommit. Waarom dit werkt is het onderwerp van het volgende gedeelte.

Versiecontrole betekent dat u kunt relaxen

Het feit dat de broncode van het project onder versiebeheer wordt gehouden, betekent dat de meeste beslissingen makkelijk ongedaan gemaakt kunnen worden. Dit gebeurt meestal als iemand een verandering commit in de veronderstelling dat iedereen daar blij mee is, en achteraf blijkt dat er bezwaren tegen bestaan. Kenmerkend voor zulke bezwaren is dat ze meestal beginnen met een verontschuldiging voor het missen van de voorafgaande discussie. Dit blijft echter achterwege als degene die het bezwaar maakt in de archieven van de mailinglist geen sporen van een

dergelijke discussie vindt. Er is hoe dan ook geen reden om, nadat de verandering gecommit is, de inhoudelijke discussie op een andere toon te voeren dan daarvoor. Iedere verandering kan worden teruggedraaid, in ieder geval tot het moment dat daarvan afhangende nieuwe veranderingen worden ingevoerd (d.w.z. nieuwe code die niet meer zou werken als die eerdere verandering zou verdwijnen). Het versiebeheersysteem geeft het project de mogelijkheid om de gevolgen van verkeerde of haastig genomen beslissingen ongedaan te maken. Dit geeft mensen op hun beurt de vrijheid om op hun instinct te vertrouwen ten aanzien van de hoeveelheid feedback die nodig is voordat ze iets doen.

Het betekent ook dat het proces om consensus te verkrijgen niet al te formeel hoeft te zijn. Binnen de meeste projecten wordt dit op gevoel gedaan. Minimale veranderingen kunnen zonder discussie geaccepteerd worden, of na een minimale discussie gevolgd door enkele instemmende knikjes. Bij meer ingrijpende veranderingen, vooral als die veel code kunnen destabiliseren, moet men een dag of twee wachten voordat aangenomen kan worden dat er consensus is. Niemand mag natuurlijk buiten een belangrijke discussie gehouden worden alleen omdat hij zijn e-mail niet vaak genoeg heeft gelezen.

Wanneer iemand er dus zeker van is dat hij weet wat er gedaan moet worden, dan moet hij dat gewoon doen. Dit geldt niet alleen voor softwarefixes, maar ook voor updates van websites, veranderingen in documenten en verder alles waarvan aangenomen mag worden dat het niet controversieel is. Normaal gesproken komt het maar zelden voor dat een actie ongedaan gemaakt moet worden en deze gevallen kunnen van geval tot geval behandeld worden. Het spreekt vanzelf dat u mensen niet moet aanmoedigen om koppig te zijn. Er is altijd een psychologisch verschil tussen een besluit waarover nog gediscussieerd wordt en een besluit dat al is doorgevoerd, ook al is het technisch terug te draaien. Mensen hebben altijd het gevoel dat daadkracht en actie hand in hand gaan en zullen een verandering liever willen voorkomen dan hem later terugdraaien. Als een ontwikkelaar misbruik maakt van dit feit door potentieel controversiële veranderingen te snel te committen, kunnen en moeten mensen hierover klagen en moet deze ontwikkelaar strenger worden gecontroleerd totdat er verbetering zichtbaar is.

Als geen consensus bereikt wordt, stem dan

Het is niet te vermijden dat in sommige discussies geen consensus bereikt wordt. Als alle andere manieren om een patstelling op te lossen falen, dan moet er gestemd worden. Maar voordat er kan worden gestemd, moet er een aantal duidelijke keuzemogelijkheden op het stembiljet worden gezet. Ook hier loopt het normale proces van de technische discussie op vloeiende wijze over in de beslissingsprocedures van het project. Het soort vragen waarover gestemd moet worden betreft vaak complexe zaken met vele facetten. In dergelijke complexe discussies zijn er gewoonlijk een of twee mensen die de rol spelen van *eerlijke bemiddelaars.* Ze posten regelmatig samenvattingen van de verschillende argumenten en houden de belangrijkste punten van onenigheid (en overeenstemming) bij. Dankzij deze samenvattingen heeft iedereen een beter idee van de voortgang die is geboekt en welke zaken nog aandacht behoeven. De samenvattingen kunnen ook als prototype dienen voor het stembiljet, mocht er gestemd moeten worden. Als de eerlijke bemiddelaars hun werk goed doen, kunnen ze geloofwaardig om een stemming vragen als de tijd daar is en zal de groep bereid zijn het stembiljet te gebruiken dat gebaseerd is op hun samenvattingen van de problematiek. De bemiddelaars zelf kunnen deelnemen aan de discussie. Ze hoeven niet buiten het gekrakeel te blijven, zolang ze de mening van de anderen maar begrijpen en die eerlijk weergeven, en zolang ze de discussie neutraal blijven samenvatten en niet hun eigen standpunten laten prevaleren.

De uiteindelijke inhoud van het stembiljet is gewoonlijk niet controversieel. Tegen de tijd dat er gestemd moet worden, is de onenigheid meestal tot een paar essentiële kwesties teruggebracht, met herkenbare labels en korte beschrijvingen. Af en toe protesteert een ontwikkelaar tegen het stembiljet zelf. Soms is dit bezwaar legitiem, bijvoorbeeld als een belangrijke keuze niet is opgenomen of niet correct is beschreven. Maar het komt voor dat een ontwikkelaar alleen maar het onvermijdelijke probeert af te wenden, wetende dat de stemming waarschijnlijk niet in het voordeel van zijn keuze zal uitvallen. Raadpleeg het gedeelte ‘Moeilijke mensen’ in *Hoofdstuk 6, Communicatie* om te zien hoe u om moet gaan met dit soort tegenwerking.

Vergeet niet het kiessysteem vast te leggen. Er bestaan allerlei systemen en men zou een verkeerd idee kunnen krijgen over welke procedure gevolgd wordt. Een goede keus in de meeste gevallen is *instemmingsverkiezing*, waarbij iedere kiezer voor zoveel opties op het stembiljet kan kiezen als hij wil. Instemmingsverkiezingen zijn makkelijk uit te leggen en te tellen en hebben in tegenstelling tot andere methodes maar één verkiezingsronde. Zie http://en.wikipedia.org/wiki/Voting_system#List_of_systems voor meer informatie over instemmingsverkiezingen en andere kiessystemen. Probeer echter uitgebreide discussies over het te gebruiken kiessysteem te vermijden, want voor u het weet raakt u in een discussie verzeild over welk kiessysteem te gebruiken om het kiessysteem te kiezen!). Eén van de redenen waarom een instemmingsverkiezing een goede optie is, is omdat er moeilijk iets tegenin te brengen valt: veel eerlijker dan dit kiessysteem kan bijna niet.

Voer ten slotte de stemming openbaar uit. Er is geen reden tot geheimhouding of anonimiteit in een stemming over zaken waarvan de discussie toch al publiekelijk gevoerd is. Laat iedere kiezer zijn stem posten op de mailinglijst van het project, zodat iedereen de stemming zelf kan natellen en alles in het archief opgenomen kan worden.

Wanneer te stemmen?

Het moeilijkste van verkiezingen is beslissen wanneer u ze gaat houden. In principe zou een stemming een uitzondering moeten zijn. Het is een laatste redmiddel wanneer alle andere opties gefaald hebben. Denk niet dat stemmen een geweldige manier is om een discussies op te lossen. Dat is het niet. Het maakt weliswaar een einde aan de discussie, maar daarmee ook aan het creatief denken over het probleem. Zolang de discussie gaande is, is er een mogelijkheid dat iemand met een nieuwe oplossing komt die iedereen bevalt. Verrassend genoeg gebeurt dit vaak: een levendige discussie kan tot een nieuwe manier van denken over het probleem en een voorstel leiden waarmee uiteindelijk iedereen tevreden is. Zelfs wanneer er geen nieuw voorstel komt dan is het nog steeds beter om een compromis te sluiten dan een stemming te houden. Na een compromis is *iedereen* een klein beetje onte-

vreden; na een stemming zijn sommigen blij, terwijl anderen zeer ontevreden zijn. Vanuit een politiek standpunt gezien heeft de eerste situatie de voorkeur. Daarbij kan iedereen ten minste het gevoel hebben dat hij een prijs bedongen heeft voor zijn ontevredenheid. Hij is weliswaar ontevreden, maar dat zijn alle anderen ook.

Het belangrijkste voordeel van stemmen is dat het uiteindelijk een probleem oplost, waardoor iedereen weer verder kan. Maar de oplossing komt tot stand door de koppen te tellen, in plaats door een rationele dialoog die iedereen uit doet komen bij dezelfde conclusie. Hoe ervarener mensen zijn met open source-projecten, des minder happig ze zijn om problemen op te lossen door middel van een stemming. In plaats daarvan zullen ze proberen om eerder voorgestelde oplossingen nog eens grondig te bekijken of hun eigen standpunt nog wat verder bij te stellen dan ze aanvankelijk gepland hadden om een compromis te bereiken. Er bestaan diverse technieken om een voorbarige stemming te voorkomen. Het meest voor de hand liggende is om te zeggen “Ik denk niet dat we al een stemming toe zijn” en dan uitleggen waarom niet. Een ander manier is om te vragen om een informele (niet bindende) stemming door handopsteken. Als de respons duidelijk overhelt naar een bepaalde kant, dan zijn sommigen eerder geneigd tot een compromis, waardoor een echte stemming niet meer nodig is. Het meest effectief is echter het aanbieden van een nieuwe oplossing of een nieuwe kijk op een oud idee, zodat men zich opnieuw over de kwestie buigt in plaats van alleen maar dezelfde argumenten te herhalen.

In bepaalde zeldzame gevallen is iedereen het erover eens dat alle beschikbare compromissen stuk voor stuk slechter zijn dan de afzonderlijke oplossingen zonder compromis. Wanneer dat gebeurt, valt er minder in te brengen tegen een stemming, enerzijds omdat het waarschijnlijk is dat het tot een betere oplossing leidt en anderzijds omdat mensen, ongeacht de uitkomst, niet bijzonder ontevreden zullen zijn. Maar zelfs dan mag de stemming nooit overhaast plaatsvinden. De discussie die plaatsvindt in de aanloop naar de stemming is hetgeen waarvan de stemmers leren. Die discussie vroegtijdig stoppen komt de kwaliteit van het resultaat niet ten goede.

(NB: Het advies om liever geen stemming te houden, gaat niet op voor het stemmen over het opnemen van een verandering zoals beschreven in het gedeelte ‘Een release stabiliseren’ in Hoofdstuk 7*, Downloadpakketten maken, releases en dagelijkse ontwikkeling*. In dat geval is stemmen meer een communicatiemechanisme, een manier om iemands betrokkenheid te registreren in het proces van het evalueren van veranderingen, zodat iedereen kan zien hoeveel feedback een bepaalde verandering ontvangen heeft.)

Wie mag er stemmen?

Als u eenmaal een kiessysteem hebt gekozen, dient de kwestie van het electoraat zich aan: wie mag er stemmen? Deze vraag ligt gevoelig, omdat op dit moment officieel erkend moet worden dat binnen het project sommigen meer betrokken zijn of een betere beoordeling kunnen maken dan anderen.

De beste oplossing is om een al bestaand onderscheid te nemen, namelijk dat voor commit access en daaraan stemprivileges toe te voegen. In projecten waar zowel gehele als gedeeltelijke toegang mogelijk is, hangt de vraag of mensen met een gedeeltelijke toegang mogen stemmen grotendeels af van het proces waarmee de toegang verleend wordt. Als het project deze toegang makkelijk verleent, bijvoorbeeld als een manier om de vele door derden aangedragen tools in de repository te onderhouden, dan zou duidelijk gemaakt moeten worden dat de gedeeltelijke toegang alleen geldt voor het aandragen van veranderingen en niet het stemmen. Omgekeerd is dit natuurlijk eveneens van toepassing: omdat volwaardige committers stemprivileges *zullen* hebben, moeten ze niet alleen als programmeurs gekozen worden maar ook als stemgerechtigden. Als iemand op de mailinglijst verstorend of dwars gedrag vertoont, dan moet de groep zeer terughoudend zijn in het benoemen van deze persoon tot committer, ook al is hij technisch bekwaam.

Het kiezen van nieuwe committers – zowel met volledige als gedeeltelijke toegang - zou eigenlijk ook met het kiessysteem moeten gebeuren. Maar dit is een van die zeldzame gevallen waarbij geheimhouding gewenst is. U kunt niet op een publieke mailinglijst stemmen over potentiële committers, omdat de kandidaat gekwetst en zijn reputatie geschaad kan worden. In plaats daarvan is het gebruikelijk dat een committer een voorstel om iemand anders commit-toegang te verlenen op een mailinglijst post waartoe alleen andere committers toegang hebben. De andere committers kunnen vrijuit spreken, omdat ze weten dat de discussie privé is. Vaak is er geen verschil van mening en is er dus geen stemming nodig. Na een paar dagen gewacht te hebben om er zeker van te zijn dat iedere committer de kans heeft gehad te reageren, zendt degene die het voorstel gedaan heeft de kandidaat bericht en biedt hem volledige toegang aan. Als er wel verschil van mening is dan volgt er, zoals voor elke kwestie, een discussie, die kan uitmonden in een stemming. Om dit proces eerlijk en open te kunnen laten verlopen, zou het feit dat erover wordt gediscussieerd al geheim moeten zijn. Als de desbetreffende persoon zou weten wat er gaande was en daarna geen volledige toegang aangeboden zou krijgen, zou hij kunnen concluderen dat hij de stemming verloren heeft en zou hij zich waarschijnlijk gekwetst voelen. Als iemand nadrukkelijk om volledige toegang vraagt, is er uiteraard geen andere keus dan het voorstel te overwegen en iemand expliciet te accepteren of af te wijzen. Als dat laatste het geval is, dan moet dat zo beleefd mogelijk gebeuren en met een duidelijke uitleg: “We stellen je patches zeer op prijs, maar we hebben er nog niet genoeg gezien om te kunnen oordelen”, of “We waarderen al je patches, maar ze hadden nog aanzienlijke aanpassingen nodig voordat ze in gebruik genomen konden worden. We zijn dus nog niet voldoende gerustgesteld om je volledige toegang te geven. We hopen dat dit na verloop van tijd zal veranderen”. Vergeet niet dat wat u zegt een klap in iemands gezicht kan zijn, afhankelijk van het zelfvertrouwen van een persoon. Probeer het bericht te schrijven vanuit zijn perspectief.

Omdat het toevoegen van een nieuwe committer meer consequenties heeft dan de meeste andere eenmalige beslissingen, hebben sommige projecten speciale voorwaarden voor de stemming. Een voorstel moet bijvoorbeeld minimaal *n* positieve stemmen krijgen en geen negatieve, of een grote meerderheid moet voor een voorstel stemmen. De exacte parameters doen er niet toe. Het belangrijkste is dat de groep voorzichtig moet zijn met het toelaten van nieuwe committers. Gelijke, of zelfs strengere bijzondere voorwaarden kunnen van toepassing zijn op stemmingen

om een committer te *verwijderen*, hoewel dit hopelijk nooit nodig zal zijn. Zie het gedeelte 'Committers' in Hoofdstuk 8*, Het managen van vrijwilligers* voor meer informatie over de niet aan het stemmen gerelateerde aspecten van het toevoegen of verwijderen van committers.

Opinie peilen versus stemmen
Voor bepaalde stemmingen kan het nuttig zijn het aantal stemgerechtigden uit te breiden. Als de ontwikkelaars bijvoorbeeld niet kunnen bepalen of een bepaalde keus voor een interface past bij de manier waarop mensen daadwerkelijk de software gebruiken, kan het een oplossing zijn om aan alle deelnemers van de mailinglijst van het project te vragen om te stemmen. Dit is eigenlijk meer een *peiling* dan een stemming, maar de ontwikkelaars kunnen ervoor kiezen om het resultaat als bindend te beschouwen. Zoals met elke peiling moet u aan de deelnemers duidelijk maken dat ze kunnen reageren. Als iemand met een beter idee komt dan wat in de vragen van de peiling staat, dan kan deze respons achteraf het belangrijkste resultaat van de peiling blijken te zijn.

Veto's
Sommige projecten kennen een vetorecht. Een veto is een mogelijkheid voor een ontwikkelaar om een haastige of niet goed overwogen verandering een halt toe te roepen, of deze in ieder geval zo lang te vertragen dat men er over kan discussiëren. Een veto zit tussen een zeer sterk bezwaar en obstructie in. De exacte betekenis ervan varieert van project tot project. Binnen sommige projecten is het erg moeilijk een veto terzijde te schuiven, bij andere kan een veto ongeldig worden verklaard met een meerderheid van stemmen, eventueel na een geforceerd uitstel om verder te kunnen discussiëren. Ieder veto moet vergezeld gaan van een grondige uitleg. Een veto zonder zo'n uitleg zou meteen ongeldig moeten worden verklaard.

Vetorecht leidt altijd tot de mogelijkheid van vetomisbruik. Soms willen ontwikkelaars door middel van een veto de druk verhogen, terwijl er eigenlijk alleen behoefte bestond aan meer discussie. U kunt misbruik van veto's voorkomen door zelf terughoudend te zijn met het uitspreken van veto's en door het vriendelijk te melden wanneer iemand zijn vetorecht te vaak gebruikt. Indien nodig kunt u de groep er ook aan herinneren dat veto's alleen bindend zijn zolang de groep bepaalt dat ze dat zijn. Als een duidelijke meerderheid van de ontwikkelaars uiteindelijk X wil, dan is X wat hoe dan ook gebeurt. Of de ontwikkelaar die het veto uitsprak trekt dit terug, of de groep besluit het gewicht van het veto te verminderen.

Het komt voor dat mensen '-1' schrijven om hun veto uit te spreken. Dit gebruik komt van de Apache Software Foundation, die gebruik maakt van een zeer gestructureerd stem- en vetoproces. Het staat beschreven op http://www.apache.org/foundation/voting.html. De Apache-normen hebben zich verspreid over andere projecten en u kunt zien dat hun afspraken, in verschillende gradaties, gebruikt worden op veel plaatsen in de open source-wereld. Technisch gezien betekent '-1' niet altijd een formeel veto, zelfs niet volgens de Apache-normen, maar informeel wordt het meestal opgevat als een veto, of op zijn minst als een zeer sterk bezwaar ergens tegen.

Net zoals stemmen kunnen veto's met terugwerkende kracht gelden.
Het is niet correct om tegen een veto bezwaar te maken op grond van het feit dat de aanpassing in kwestie al toegepast is of de betreffende actie al ondernomen (tenzij het iets is dat niet terug te draaien is, zoals een persbericht).

Aan de andere kant is de kans dat een veto dat weken later uitgesproken wordt serieus wordt genomen erg klein; en zo hoort het ook.

4.3 ALLES OPSCHRIJVEN

Op een gegeven moment wordt het aantal afspraken en overeenkomsten binnen het project zo groot dat u ze ergens vast moet leggen. Om zo'n document geldigheid te geven moet u duidelijk maken dat het gebaseerd is op discussies uit de mailinglijst en op besluiten die al in werking zijn getreden. Refereer bij het samenstellen van dit document aan de betreffende threads in de archieven van de mailinglijst. Wanneer er een punt is waar u niet zeker van bent, vraag het dan opnieuw. Het document mag niet voor verassingen zorgen. Het is niet het uitgangspunt van de afspraken maar slechts een beschrijving ervan. Als het succesvol is, is het natuurlijk wel zo dat mensen het zullen citeren alsof het een bron met autoriteit is. Dat betekent eigenlijk alleen maar dat het de globale mening van de groep goed weergeeft.

Dit is het document waaraan gerefereerd wordt in het gedeelte 'Ontwikkelaarsrichtlijnen' in Hoofdstuk *2, Aan de gang* Wanneer het project zich nog in de beginfase bevindt, kunt u bij het bepalen van de richtlijnen niet terugvallen op een lange projectgeschiedenis. Naarmate de ontwikkelaarsgemeenschap echter groeit en volwassener wordt, kunt u de tekst aanpassen aan de hand van de feitelijke gang van zaken.

Probeer niet alles op te schrijven. Geen enkel document kan alles bevatten wat mensen moeten weten over hoe ze deel moeten nemen aan een project. Veel van de afspraken die betrekking hebben op een project blijven altijd onuitgesproken en worden nooit expliciet benoemd, maar worden toch door iedereen nageleefd. Andere zaken zijn gewoon zo overduidelijk dat ze niet genoemd hoeven worden en alleen maar afleiden van de belangrijke, minder voor de hand liggende zaken. Het heeft bijvoorbeeld geen zin richtlijnen te schrijven als "Wees beleefd en respectvol naar anderen op de mailinglijst en houd u verre van vuilbekkerij" of "Schrijf goede en leesbare codes zonder bugs". Natuurlijk zijn dit allemaal wenselijke zaken maar omdat er geen universum denkbaar is waarin dit *niet* zo is heeft het geen zin ze te noemen. Als mensen onbeleefd zijn op de mailinglijst of een code vol bugs schrijven, dan zullen ze daar niet mee stoppen omdat de richtlijnen van het project dat zeggen. Dergelijke situaties moeten opgelost worden op het moment dat ze de kop opsteken en niet door algemene aansporingen om braaf te zijn. Aan de andere kant, als het project specifieke richtlijnen heeft *hoe* een goede code moet worden geschreven, bijvoorbeeld regels om iedere API te documenteren in een bepaalde format, dan moeten deze richtlijnen zo compleet mogelijk vastgelegd worden.

Een goede manier om te bepalen wat u op moet nemen in dit document is om het te baseren op de vragen die nieuwkomers het meest stellen en op de klachten die ervaren ontwikkelaars het vaakst uiten. Dit houdt niet automatisch in dat dit document een FAQ-pagina wordt. Het moet waarschijnlijk een meer consistente en beschrijvende structuur hebben dan een FAQ kan bieden. Ook in dit document zou de aanpak van de kwesties echter op de realiteit gebaseerd moet zijn, in plaats van te anticiperen op de kwesties zie zich eventueel zouden kunnen voordoen.

Als het project geleid wordt door een VD of als er mensen zijn met speciale bevoegdheden (manager, voorzitter, wat dan ook), dan biedt dit document tevens een goede gelegenheid om de opvolgingsprocedures vast te leggen. Soms kan dit zo simpel zijn als een lijstje met namen van mensen die de VD kunnen vervangen, mocht hij het project om wat voor reden dan ook verlaten. Over het algemeen kan alleen de VD een opvolger benoemen. Als er gekozen verantwoordelijken zijn, dan moet de procedure hoe deze mensen zijn genomineerd en gekozen beschreven worden in het document. Als er aanvankelijk geen procedure was, dan moet er consensus worden bereikt over een procedure in de mailinglijst *voordat* u erover schrijft. Mensen kunnen soms lange tenen hebben als het om hiërarchische structuren gaat. Daarom moet hier omzichtig mee worden omgegaan.

Misschien is het wel het belangrijkst om duidelijk te maken dat regels heroverwogen kunnen worden. Als de afspraken zoals beschreven in het document het project gaan hinderen, herinner iedereen er dan aan dat het verondersteld wordt een reflectie te zijn van de intenties van de groep en geen bron van frustratie en blokkades. Als iemand er een gewoonte van maakt om de regels iedere keer dat ze hem in de weg staan te willen herzien, dan hoeft u hier niet altijd met deze persoon over in discussie te gaan. Soms is zwijgen de beste tactiek. Als andere mensen het eens zijn met de klachten, dan zullen ze de klager bijvallen en is het duidelijk dat er iets moet veranderen. Maar als niemand het met hem eens is, dan zal de klager weinig respons krijgen en blijven de regels zoals ze zijn.

Twee goede voorbeelden van richtlijnen binnen een project zijn het Subversion-bestand `hacking.html`, op http://svn.collab.net/repos/svn/trunk/www/hacking.html en de governance-documenten van de Apache Software Foundation, op http://www.apache.org/foundation/how-it-works.html en http://www.apache.org/foundation/voting.html. De ASF is in feite een verzameling van softwareprojecten, wettelijk georganiseerd als een organisatie zonder winstoogmerk. De bijhorende documenten beschrijven dus meer de governance-procedures dan de afspraken over de ontwikkeling van programmatuur. Maar ze blijven de moeite van het lezen waard, omdat ze de verzamelde ervaringen van veel open source-projecten vertegenwoordigen.