De Open Source Software Architect

Bert Dingemans

Door het overheidsbeleid omtrent open source software (OSS) en open standaarden wordt van (overheids)organisaties verwacht dat een implementatiestrategie voor OSS ontwikkeld wordt. Open source software kan kostenbesparend werken en stimuleert de lokale economie is de macro economische opstelling. Over open source bestaan veel mythen waarvan in dit artikel er een aantal beschreven worden. Echter bij de selectie van een software component zal een architect een gewogen advies moeten kunnen maken. Dit artikel beschrijft een aantal toetsingsmogelijkheden voor de gelijke geschiktheid van zowel open als gesloten software: kosten, juridisch, volwassenheid en non functionals. Daarnaast wordt kort ingegaan op de risico's van complexiteitstoename door verandering in een applicatielandschap.

Inleiding

Open source software (OSS) heeft zich de laatste jaren een plek verworven in de softwarewereld. Kijk je bijvoorbeeld op www.osalt.com (een vergelijkingssite van open en gesloten software) dan zie je bij steeds meer categorieën open source varianten beschikbaar komen.

Open source software is daarmee een serieus alternatief geworden voor de veel gebruikte toepassingen binnen applicatielandschappen. Toch zie je dat er nog maar mondjesmaat gekozen wordt voor open source alternatieven. Vooral binnen architectuurafdelingen zie je in het meest positieve geval twijfel over OSS alternatieven, niet zelden is er zelfs afkeer voor OSS oplossingen.

Persoonlijk denk ik dat OSS een blijvertje is en op meer en meer vlakken toegepast zal worden. Dat betekent dat (software) architecten in hun inventarisaties en adviezen OSS moeten gaan meewegen. Er zal een afgewogen oordeel moeten komen tussen gesloten en open alternatieven. Niet in de laatste plaats omdat het OSS perspectief ook bij afnemers meer en meer op de agenda staat. Er wordt in mijn dagelijkse werk regelmatig door de klant expliciet gevraagd om OSS alternatieven mee te nemen in de afweging.

In de afgelopen periode ben ik betrokken geweest bij het opstellen van een OSS strategie voor een ministerie. Binnen deze strategie is onderzocht welke randvoorwaarden gesteld moeten worden aan de introductie van OSS. Belangrijkste onderdeel hiervan is het beheersen van risico's. Dit document beschrijft vanuit de ervaringen in bovengenoemde activiteiten. Naast de vele mythen die ik gehoord heb wordt ingegaan op een model om de gelijke geschiktheid van alternatieven te toetsen.

Wat is open source software

De NOiV site geeft een heldere definitie van open source software, deze luidt als volgt:

Open source software is software met twee kenmerken:

- De broncode van de software is vrij beschikbaar.
- In het licentiemodel is het intellectueel eigendom en het (her)gebruik van de software en bijbehorende broncode dusdanig geregeld dat de licentienemer de broncode mag inzien, gebruiken, verbeteren, aanvullen en distribueren.

Een open source licentie dwingt af dat de broncode van het product, en soms ook de aanpassingen daarop, vrij beschikbaar moet zijn. Om helder te maken wanneer software open source software genoemd mag worden, zijn door de Open Source Initiative voorwaarden opgesteld waaraan een licentie moet voldoen, zodat de software die onder die licentie vrijgegeven wordt, open source genoemd mag worden.

Overheidsbeleid

Binnen de Nederlandse overheid spelen open source software en open standaarden een steeds belangrijkere rol. In 2002 is door Kees Vendrik een motie in de tweede kamer ingediend waarmee het gebruik van OSS ingezet moest worden om de Nederlandse ICT markt te stimuleren.

In 2007 is op basis van deze motie het actieplan “Nederland Open in Verbinding” aangenomen door de ministeries van BZK en EZ. Hiermee wordt een concrete invulling gegeven aan het stimuleren van overheidsorganisaties om open source software te gebruiken. Echter het actieplan stelt OSS niet als norm. Binnen de overheid geldt dat open source software de voorkeur heeft boven gesloten source software, als beide soorten software verder aan dezelfde eisen voldoen.

11 mythen over Open Source Software

Over OSS bestaan veel mythen, zowel door voor- als tegenstanders worden deze mythen ingezet in de, veelal emotionele, discussies over dit onderwerp. In onderstaande opsomming tien mythen over OSS en één mythe over het Nederlandse overheidsbeleid, inclusief een toelichting op de mythen.

1 Met open source software is het beheer niet geregeld waardoor ik een continuïteitsrisico loop

Beheer van software heeft weinig met open of gesloten software te maken. Vaak wordt er bedoeld dat er geen partijen zijn te vinden die voor het beheer benaderd kunnen worden. Dit heeft met de volwassenheid van de product organisatie te maken en zal voor zowel open als gesloten software de ene keer beter geregeld zijn dan de andere.

2 Open source software is minder volwassen dan gesloten software

Volwassenheid heeft alles te maken met de organisatie rondom een product. Argument wordt vaak aangevoerd met een vergelijking van twee producten die niet met elkaar te vergelijken zijn. Bij zowel open als gesloten software is het bij selectie van een product goed te kijken naar de volwassenheid van het product.

3 Open source software wordt ontwikkeld door niet aan te sturen idealistische nerds.

Van een fervent tegenstander van open source kreeg ik eens een mailtje met de volgende bijlage als voorbeeld van een OSS programmeur.

Ook hierbij geldt dat open source en gesloten source niet het uitgangspunt is. Veelal dragen professionele ontwikkelaars onder werktijd bij aan de ontwikkelingen binnen een open source community. Meestal met redenen om concurrentievoordeel te bereiken op een product dat potentie heeft in de markt.

4 De grote systeem integrators (waar wij mee samenwerken) werken niet met open source software

Deze stelling is niet waar, de grote system integrators zijn steeds beter in staat om open source oplossingen aan te bieden. Bij een aanbestedingstraject heb ik gezien dat door te vragen om een open source alternatief bij gelijke geschiktheid vier van de vijf aanbieders in staat waren een goed open source alternatief aan te bieden.

5 Met open source software zijn de informatiebeveiligingsrisico's groter

Informatiebeveiliging is zowel bij gesloten als open source een onderwerp dat aandacht verdiend. Echter deze mythe over open source gaat niet op. Doordat de programma code door iedereen bekeken kan worden zijn beveiligingsrisico's snel op te sporen en op te lossen. Daarnaast zijn veelal heel veel verschillende builds van de software beschikbaar wat het ontwikkelen van bijvoorbeeld virussen minder interessant is voor OSS.

6 Ontwikkelaars en beheerders willen niet werken met open source software

Voor een individuele ontwikkelaar kan dit gelden, over het algemeen zie je dat ontwikkelaars en beheerders al regelmatig zelf kijken naar open source oplossingen in de vorm van libraries en hulpmiddelen. Denk aan producten als Hibernate en Jboss die met name door de acceptatie bij de ontwikkelaars een enorme opgang hebben gemaakt.

7 Open source software is van niemand wat juridische risico's met zich meebrengt.

Is niet waar, open source software is altijd van iemand (veelal een organisatie). De broncode is vrijgegeven maar de eigenaar kan aanvullende eisen stellen hoe hiermee omgegaan wordt. Dit is een aandachtspunt bij het selecteren van een open source oplossing, zie hiervoor de paragraaf over juridische toetsing. Ander aandachtspunt is overnames van bedrijven die open source producten in hun portfolio hebben. Dit brengt een onzekerheid met zich mee, echter ook bedrijven met gesloten software veranderen door overnames. Wederom geen verschil tussen open en gesloten software.

8 Open source software is meestal gratis of ten minste goedkoper

Voor de licenties hoeft niet betaald te worden bij open source, dat neemt niet weg dat de andere diensten bij een product geen geld kosten. Daarnaast is soms een deel van het product-portfolio open source maar zijn een aantal essentiële componenten gesloten source. Als laatste zal men in een aantal gevallen expertise van het OSS product moeten inkopen.

9 Aan open source software verdienen leveranciers niets

Hiervoor geldt ongeveer hetzelfde als bij bovenstaand punt. Veel leveranciers hebben diensten ontwikkeld rond open source producten en hierop hun verdienmodel ingericht. Aandachtspunt is dat licentieinkomsten ingezet worden voor sales en presales activiteiten binnen gesloten software. Hierin heeft open source echt een achterstand bijvoorbeeld bij aanbestedingstrajecten.

10 Bij open source software heb ik geen invloed op toekomstige ontwikkelingen van het product

Ook dit hangt af van de volwassenheid van de organisatie rondom open en gesloten source software. Hierbij zijn grote verschillen te zien, denk bijvoorbeeld aan een product als DotNetNuke een open source CMS waarvoor wereldwijd conferenties worden georganiseerd. Bij een gelijke geschiktheidtoets is gezien de grote verschillen dit veelal een belangrijk punt van onderscheid maar is dit onderscheid veelal niet gebaseerd op open of gesloten source aspecten.

11 Voor OSS geldt binnen de overheid het "pas toe of leg uit" principe

Dit is een hardnekkig misverstand! Ontstaan vanwege het feit dat dit principe geldt voor open standaarden en open source veelal hiermee in verband wordt gebracht. Overheidsbeleid is gericht op het principe "bij gelijke geschiktheid verdient open source de voorkeur".

Bovenstaande mythen geven aan dat er nog veel onduidelijk is over open source. Dit neemt niet weg dat bij het ontwikkelen van een oplossing een architect een gewogen beslissing dient te nemen. In de volgende paragraaf laten we een voorbeeld zien hoe dit er uit kan zien.

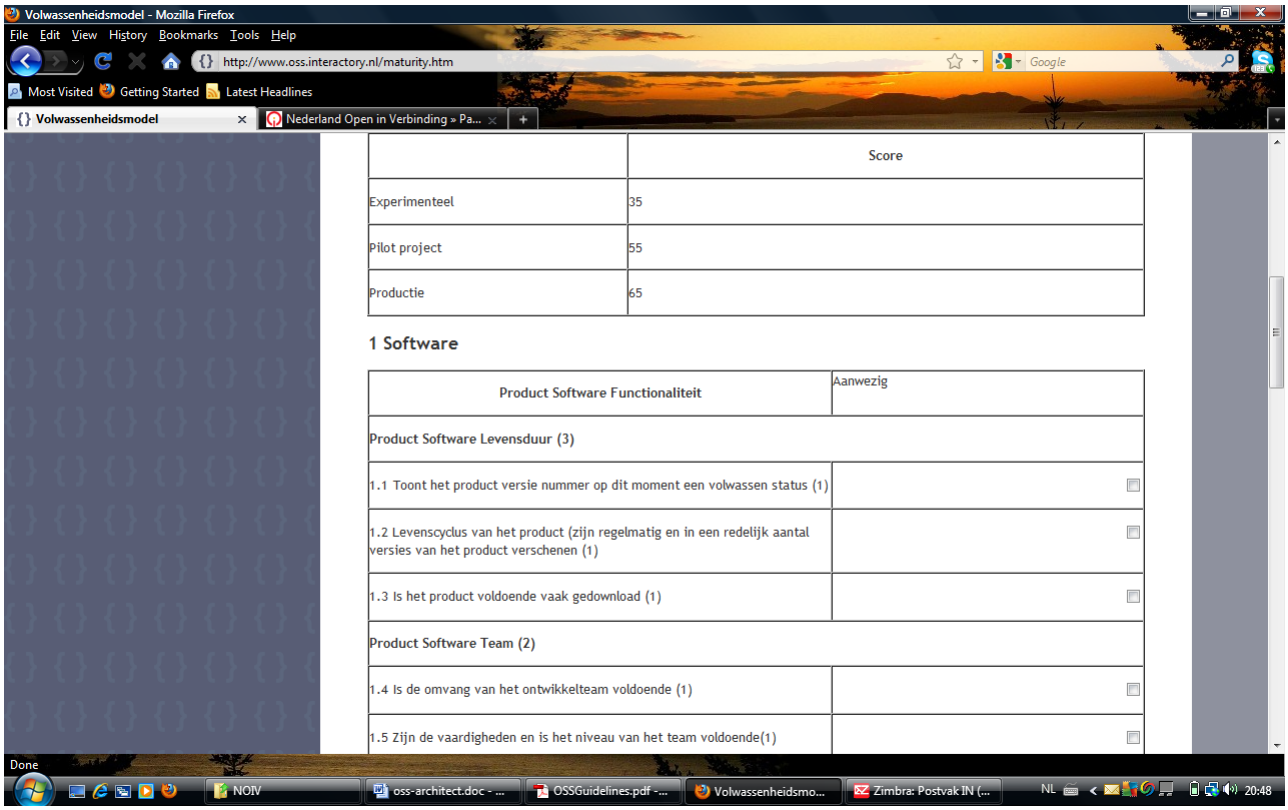
Toetsing gelijke geschiktheid

Voor Nederlandse overheden geldt dat bij gelijke geschiktheid gekozen moet worden voor open source oplossingen. Echter wanneer zijn twee producten “gelijk geschikt”. Het is een rekbaar begrip en in combinatie met de inzet van één of meer mythen ontstaan in een organisatie al snel een ondoorzichtig open source beleid.

Voor (software)architecten is het daarom van belang te zoeken naar mogelijkheden om gelijke geschiktheid op basis van kwantificeerbare elementen te gaan uitwerken. In de volgende paragrafen wordt dit gedaan voor een viertal aandachtsgebieden. Als laatste wordt een vijfde aandachtsgebied dat qua kwantificeerbaarheid lastig uit te werken is maar volgens mij een kritische faalfactor is, namelijk beheer- en migratieaspecten.

Volwassenheid

Deze toetsing is het eenvoudigst uitvoerbaar en meest onderscheidend. Belangrijk hierbij is dat niet alleen het softwareproduct zelf wordt getoetst maar met name de organisatiegraad om het product heen. Veelal worden volwassenheidstests alleen in verband met open source producten genoemd. Echter ook voor gesloten source is een volwassenheidstoets in veel gevallen op zijn plaats.



Figuur 1 Voorbeeldvragen toets

De toetsing bestaat uit een aantal vragen over onderwerpen als:

- Software
- Technische ondersteuning
- Documentatie
- Training
- Integratie
- Professionele ondersteuning

Veelal bestaat de toetsing uit rond de twintig vragen, waarbij de vragen met ja of nee beantwoord worden. Per vraag kan een weging opgegeven worden en de optelsom van alle positief beantwoorde vragen geeft een totaalwaarde. Per toepassing, zoals experimenteel, pilot of productie kan een grenswaarde bepaald worden. Boven deze grenswaarde kan het product ingezet worden als oplossing.

De volwassenheidstoets is volgens mij de minimale toetsing die uitgevoerd kan worden bij minder omvangrijke software selectie trajecten. De toetsing is ook relatief eenvoudig uitvoerbaar en zal veelal niet meer dan een dagdeel aan voorbereiding en uitvoering kosten.

Licenties en juridische aspecten

Voor gesloten software is het licentiebeleid van de leverancier in veel gevallen ondoorzichtig te noemen. Wie wel eens de licentie overeenkomst heeft geprobeerd te doorgronden kan beamen dat de leverancier zoveel mogelijk risico's verplaatst naar de gebruiker.

Voor open source software is een onderzoek naar de licentiestructuur noodzakelijk aangezien deze een aantal extra aspecten heeft. Met name het doen van aanpassingen in de broncode van een product en het beleid hoe hiermee omgegaan wordt dient onderzocht te worden.

Belangrijk onderwerp hierin is copyleft. Wikipedia schrijft het volgende over copyleft:

Copyleft is de toepassing van de auteursrechtwetgeving om het publiek de vrijheid te geven om een creatie en alle daarvan afgeleide werken te wijzigen, te verbeteren en te herdistribueren. Copyleft kan hierbij gezien worden als een parodie op *copyright* (het woord *right* betekent in het Engels naast 'recht' tevens 'rechts', en 'left' verwijst dan naar het tegenovergestelde: links), waarbij men een bepaalde creatie afschermt van vrije distributie.

Voor het bepalen welke licentievoorwaarden en de gevolgen voor een organisatie zijn een tweetal websites te gebruiken met een gelijke geschiktheidstoets, te weten:

- https://noiv.nl/open-source-licentiewijzer/
- http://www.opensource.org/licenses/index.html

Uitvoeren van de licentie toets op basis van de NOiV licentiewijzer is eenvoudig uitvoerbaar en vergt een beperkte inzet. Daarnaast verdient het de voorkeur de licentietekst van het specifieke product te toetsen bij een jurist.

Licenties kunnen in sommige gevallen voor problemen zorgen. In een traject, waarbij ik betrokken ben geweest, was een aanvullende licentievoorwaarde opgenomen dat de leverancier gebruik moest goedkeuren. Echter hoe die goedkeuring verkregen moest worden was volstrekt onduidelijk. Ook mails met vragen hierover bleven onbeantwoord. Vanuit het oogpunt van zorgvuldigheid is verdere inventarisatie van dit product gestaakt.

Non functionele requirements

Een toetsing die ingezet kan worden maar hogere eisen stelt aan de voorbereiding van de toetsing heeft betrekking op de non functionele requirements. Op basis van het Quint 2 model kunnen non functionele requirements ingedeeld worden in de Quint 2 rubrieken. Vervolgens kan per rubriek een wegingsfactor opgegeven worden.

De vragen in de rubriek worden vervolgens per software product gescoord in welke mate voldaan wordt aan de eisen zoals geformuleerd. Op basis van een totaal van de wegingsfactor en de score van alle rubrieken is het mogelijk om twee of meer producten op basis van dit Quint 2 model met elkaar te vergelijken. In de voorbeeldapplicatie is een voorbeeld opgenomen van deze opzet met behulp van Quint 2.

De non functionele requirements toetsing is ten opzichte van de volwassenheidstoets complexer van opzet. Enerzijds omdat bepaald moet werken per non functional welke eisen er precies gesteld worden voor de weging. Dat vergt een onderzoek vooraf naar de non functionele aspecten van software producten die nog niet aangeschaft zijn. Vervolgens moeten de producten op basis van de gestelde eisen getoetst worden.

Houdt hierbij rekening met het feit dat niet alle functionele requirements voor alle softwaregebieden relevant zijn. Ook is het mogelijk om alleen de meest relevante non functionele eisen in deze toetsing te wegen. Dit is de reden waarom in deze toetsing altijd twee producten met elkaar vergeleken worden en er geen grenswaarden bepaald zijn zoals bij de volwassenheidstoets.

Kosten

Kosten van software wordt vaak ten voordele van open source genoemd, er hoeft ten slotte niet betaald te worden voor licenties. Toch dient hier een weging gemaakt te worden van de gelijke geschiktheid. Hiertoe kunnen de volgende kosten in kaart gebracht te worden:

- Ontwikkel- en inrichtingkosten
- Migratiekosten
- Operationele kosten

In de voorbeeldapplicatie is het kostenmodel uitgewerkt op basis van een tarievenboek, dit maakt het mogelijk om een globale uitwerking van de kosten op eenvoudige wijze te bepalen. Voor het bepalen van een software architectuur is dit meestal voldoende. Andere modellen zoals TCO kunnen een extra detaillering geven, met name voor generieke voorzieningen. Echter deze berekening wordt al snel complex.

Bij de toetsing dienen altijd twee producten met elkaar vergeleken te worden. Dat heeft te maken met het feit dat gesloten software veelal licentiekosten hebben. Echter de overstap op open source brengt soms extra migratie- en inrichtingskosten met zich mee. Hierdoor is een totaalvergelijking tussen minimaal twee oplossingen noodzakelijk.

Beheer- en migratieaspecten

In de bovenstaande paragrafen is een vergelijksmodel van een viertal aspecten van software selectie gegeven. Op basis van dit model is het mogelijk om gelijke geschiktheid te bepalen. In de praktijk zijn de voorbeelden van mislukte open source implementaties echter bekend. Betekent dit dat er extra aspecten zijn die aandacht verdienen bij de software selectie traject? Ik denk dat we deze vraag met een ja kunnen beantwoorden, onder andere om de volgende redenen:

Complexiteit bij de migratie

Open source software is veelal beduidend anders opgezet dan de reeds aanwezige producten binnen een organisatie. Zeker bij overheidsorganisaties vindt men een omvangrijk applicatielandschap waarbij de onderlinge verbanden tussen de systemen omvangrijk en innig zijn en daarnaast nauwelijks in kaart zijn gebracht. Introductie van anders opgezette systemen veroorzaakt daarbij dat het applicatielandschap als een kaartenhuis in elkaar zakt en de migratie van een relatief eenvoudig onderdeel al snel omvangrijk en complex maakt. Hiermee is vooraf meestal geen rekening gehouden waarna de introductie van OSS als schuldige aangewezen wordt.

Complexiteit van beheer

Bij het beheer van componenten in een applicatielandschap is in de bestaande architectuur, ondanks de verschillende producten, toch vaak een vorm van uniformiteit ontstaan. Componenten worden beheerd in logische clusters en de indeling van de beheerteams is hierop ingericht.

Door introductie van nieuwe software componenten (open of gesloten) wordt deze uniformiteit doorbroken en zal de inrichting van zowel de clusters als de teams sterk veranderen. Deze verandering veroorzaakt dat er een periode is waarin extra inzet nodig is en dat een team zich dient te ontwikkelen om deze complexiteitstoename te gaan beheersen. Hiermee is veelal geen rekening gehouden bij de start van een implementatietraject. Gevolg is veranderingen en negatieve emoties binnen de teams.

Complexiteit door het introduceren van andersoortige software componenten (open of gesloten) is een aspect dat bij het toetsen van gelijke geschiktheid niet wordt meegewogen. Dat is een terechte keuze want producten kunnen “gelijk geschikt” zijn maar desondanks het software landschap complexer maken.

Het is gemakkelijk om de gevolgen van deze veranderingen in het landschap toe te schrijven aan het feit dat de gekozen oplossing bestaat uit open source software. Dat is niet terecht. Echter een oplossing voor dit complexiteitsprobleem is denk ik binnen huidige ICT organisaties niet direct voor handen. Mogelijk dat een complexiteitstoets of risicotoets, zoals het coramodel, voor een applicatielandschap hierbij hulp kan bieden.

OSS rekenmodel webapplicatie

In dit artikel is een prototype van het OSS rekenmodel reeds aan de orde gekomen. Het uiteindelijke prototype is gepubliceerd op internet en kan door iedereen bekeken en gebruikt worden (het is een open source product). Het geeft een beeld van de opzet van een eenvoudig rekenmodel voor toetsing van gelijke geschiktheid. Dit is te vinden op http://www.oss.interactory.nl. Momenteel wordt gewerkt aan een uitbreiding van dit rekenmodel waarbij ook beheeraspecten worden meegewogen.

Referenties

[OSOSS, 2007] OSOSS*, Het verwerven van Open Source Software*, NOiV, Den Haag, 2007.

[IDA, 2003] IDA*, The Open Source Migration Guide*, netproject, Morden Surrey, 2003.

Bert Dingemans

DLA Ontwerp & Software

bert@dla-os.nl