Ministerie van Binnenlandse Zaken en
Koninkrijksrelaties

Publieke waarden en rechten bij aanbesteden van ICT

Handreiking aanbesteden van open source software II

Over dit document; waar gaat het over?

Wie software verwerft, verwerft per definitie ook rechten. Dat kan het gebruiksrecht zijn, of ook de mogelijkheid om de software te begrijpen, te wijzigen of te delen. Sommige rechten zijn altijd noodzakelijk. Wat heb je aan software, zonder een gebruiksrecht? Andere rechten kunnen noodzakelijk zijn voor specifieke ambities. Want zonder het recht om de software in te zien en te begrijpen, geen technische transparantie. Weer andere rechten kunnen voorwaardelijk zijn om specifieke ambities op een bepaalde manier vorm te geven, zoals samenwerking, cybersecurity of verantwoording over digitale veiligheid. Maar in alle gevallen hangt het bereiken van publieke waarden – of dit nu veiligheid is, vertrouwen, rekenschap of transparantie – samen met het verwerven van eigendomsrechten. Omdat we dit doen via de inkoop in een aanbesteding, zou de rol van rechten daarom altijd een vanzelfsprekend onderdeel moeten zijn bij de verwerving van software. Want ook los van specifieke ambities rond open source software, is dit doorgaans voorwaardelijk voor het bereiken van specifieke beleidsambities via inkoop.

Dit document is een handreiking om strategisch na te denken over hoe publieke waarden vertaald kunnen worden in een stapsgewijze aanpak van een aanbesteding waar het gaat om rechten op software.

Dit document richt zich op software met directe maatschappelijke betekenis. Het gaat dan om software waaraan de overheid een deel van haar handelen delegeert, daarbij kun je denken aan (semi-) automatische besluitvorming, gegevensuitwisseling of data-analyse.

Deze publicatie is tot stand gekomen in opdracht van het Ministerie van Binnenlandse Zaken.

Disclaimer

In deze handreiking wordt niet elk scenario uitgewerkt en waar dat wel het geval is, gaat het om voorbeelden. Dit is geen (juridisch) advies, maar – hopelijk – een inspiratie om na te denken over een betere inkoopvraag.

Versie

Versie 1.1, 23 maart 2022

Inhoudsopgave

1. Samenvatting

Software die overheden gebruiken voor hun primaire proces, dus waaraan zij individuele besluitvorming delegeren, gegevensuitwisseling of data-analyse, heeft direct invloed op democratische en rechtsstatelijke waarden, maar ook op allerlei beleidsambities. Elke organisatie heeft de verantwoordelijkheid om bij de inkoop van ICT stil te staan bij de gevolgen voor de beginselen van goed bestuur en de beginselen van behoorlijk bestuur. Dit document identificeert bronnen, waarin die verantwoordelijkheid is vastgelegd. Ook zijn er meestal gevolgen voor specifieke beleidsambities en de context van de organisatie, zoals het belang van de keten. Al deze zaken hebben een directe link met de rechten die een overheid verwerft bij de inkoop van software. Dat is de kern van dit document: een bespreking van wat dit betekent voor aanbestedingen in vier scenario's.

De uitkomst van dit document zijn vier scenario's waarin verschillende eisen gesteld worden aan de rechten op software en een bespreking van wat dit betekent voor wat er in een aanbesteding kan, logischerwijs moet of strategisch handig is. De huidige situatie, waarin hiervoor weinig aandacht is, blijkt dan de meest complexe situatie.

Waarden

In hoofdstuk 4 staan een aantal waarden benoemd, die afkomstig zijn uit de beleidsbrief "Open, tenzij". Dit zijn:

- continuïteit
- leveranciersonafhankelijkheid
- marktwerking
- samenwerking en ketensamenwerking
- innovatie en bedrijvigheid
- veiligheid en verantwoording

Daaraan wordt ook transparantie toegevoegd als voorwaarde voor rechtsstatelijke waarden en met name het navolgen van het proces van besluitvorming en feitenverzameling.

Bronnen: wanneer deze waarden tevens plichten zijn

Bij een aantal van die waarden staan tevens bronnen benoemd. Dit zijn met name de waarden die samenhangen met de rechtsstatelijke beginselen van goed bestuur en behoorlijk bestuur. Deze bronnen onderbouwen waarom het doorgaans een plicht is van elke overheid om die waarden na te streven[1].

Strategische visie: organisatorische belangen én publieke belangen

Deze waarden zijn input voor de strategische visie.
De strategische visie geeft enerzijds weer hoe software moet bijdragen aan het primaire proces van de organisatie.
Dat wil zeggen hoe alle organisatorische belangen gediend worden. Anderzijds geeft de strategische visie ook weer hoe rechtsstatelijke waarden, die horen tot de opgave van elke publieke organisatie, gerealiseerd zullen worden. Dit geldt ook voor publieke waarden die volgen uit algemeen beleid, buiten de directe opgave van de eigen organisatie. Want net zoals inkoop gevolgen heeft voor klimaat en milieu en elke organisatie daarin een verantwoordelijkheid heeft, heeft elke organisatie ook de verantwoordelijkheid om stil te staan bij de gevolgen voor de rechtsstaat, haar onafhankelijkheid als overheid of het belang van de keten.

[1] Daarmee kunnen overheden de argumenten verkrijgen die bepaalde eisen aan de rechten in een aanbesteding kunnen dragen. Als er voldoende argumenten zijn om een bepaald recht te vereisen, dan is zo'n eis draagkrachtig beargumenteerd. Dat is belangrijk omdat in een aanbestedingsprocedure aanbieders ook beargumenteerd alternatieven kunnen voorstellen, die niet zonder goede motivatie kunnen worden afgewezen.

Eisen aan de rechten op software

Deze stappen leiden tot eisen aan de rechten op software in een aanbesteding. Dat komt omdat deze rechten een voorwaarde zijn voor het realiseren van die waarden. Dat wil niet zeggen dat met het vereisen van rechten die waarden gerealiseerd zijn, maar wel dat de mogelijkheid er is ze te organiseren in het proces van de ontwikkeling en het gebruik van de software van de organisatie.

Deels overlappen de zes waarden in termen van de eisen waartoe ze aanleiding geven. Daarom zijn ze samengevat in vier sets van waarden die leiden tot dezelfde eisen. Dit zijn:

1. *Waardenset 1 "rechtsstatelijke beginselen"*
   Rechtsstatelijke beginselen die leiden tot de eis om de software te kunnen begrijpen en daarmee het proces van besluitvorming en feitenverzameling.
1. Waardenset 2 "organisatorisch onafhankelijk" - omvat tevens waardenset 1 Leveranciersonafhankelijkheid bij de (verdere) ontwikkeling van software, marktwerking en continuïteit die leiden tot de eis om de software te mogen gebruiken, begrijpen, wijzigen en verspreiden
2. Waardenset 3 "frictieloze samenwerking" - omvat tevens waardenset 1 en 2
   Frictieloze samenwerking, ook bij veiligheid en verantwoording, die leiden tot de aanvullende eis van een gestandaardiseerd contract met deze ruime gebruiksrechten
3. Waardenset 4 "publiek hergebruik" - omvat tevens waardenset 1, 2 en 3
   Innovatie, bedrijvigheid, vertrouwen en transparantie die leiden tot de aanvullende eis van publieke beschikbaarheid onder een gestandaardiseerd contract met deze ruime gebruiksrechten

Van waarden naar eisen naar scenario's

Om de verschillende overwegingen bij deze waardensets zo toegankelijk mogelijk te presenteren zijn er vier scenario's geschetst.

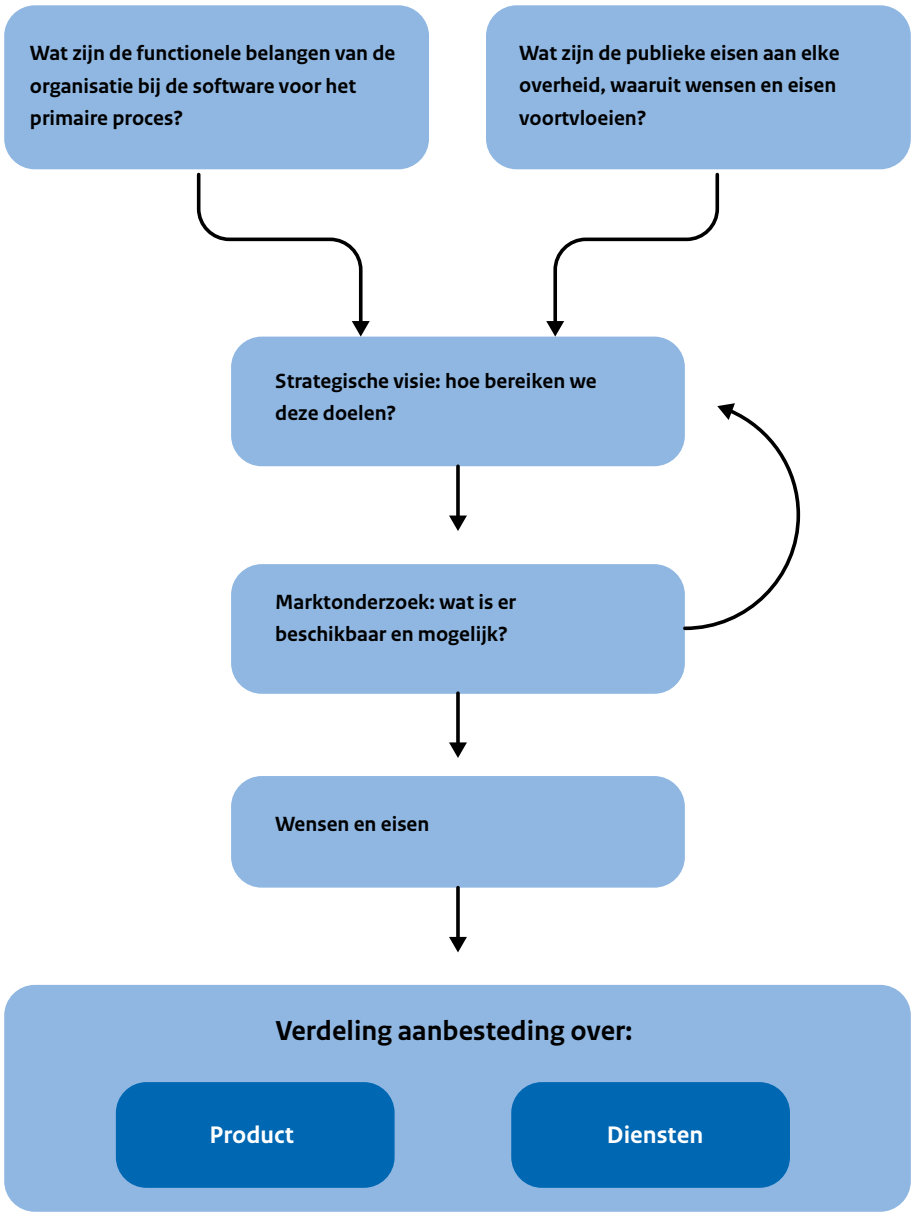
1. Scenario: de eis van een open source licentie
   De eis van een gestandaardiseerde open source licentie is het meest eenvoudige scenario. De eisen in dit scenario zijn de minimale eisen voor waardenset 3 "frictieloze samenwerking" en omvat dus ook waardenset 2 "organisatorisch onafhankelijk". Met de eis van openbare publicatie is dit tevens het scenario voor de vierde waardenset "publiek hergebruik". Dit is een relatief kleine toevoeging, die zelfs tot vereenvoudiging kan leiden.
2. Scenario: geen eisen, maar wensen
   Dit is het traditionele scenario. De keuze tussen open source software en propriëtaire software wordt via de markt bepaald. Dit is het meest complexe scenario, omdat vooraf niet duidelijk is welke rechten zullen worden verworven.
3. Scenario: de eis van inzage en distributie
   Dit scenario beantwoordt aan de eisen uit de eerste waardenset "rechtsstatelijke beginselen". Hierin verandert relatief weinig aan scenario 2 "Geen eisen, maar wensen", maar zorgt wel voor enige vereenvoudiging. In dit scenario is het beter mogelijk het softwareonderdeel "maatwerkmodule" onafhankelijk aan te besteden.
4. Scenario: de eis van gebruik, begrip, wijziging en distributie
   Dit scenario beantwoordt aan de tweede waardenset "organisatorisch onafhankelijk" maar zonder een open source licentie te vereisen. Dit scenario trekt het verschil tussen overdracht van eigendom en licensering uit elkaar en de consequenties die dit heeft voor kosten en complexiteit.

Marktonderzoek is noodzaak

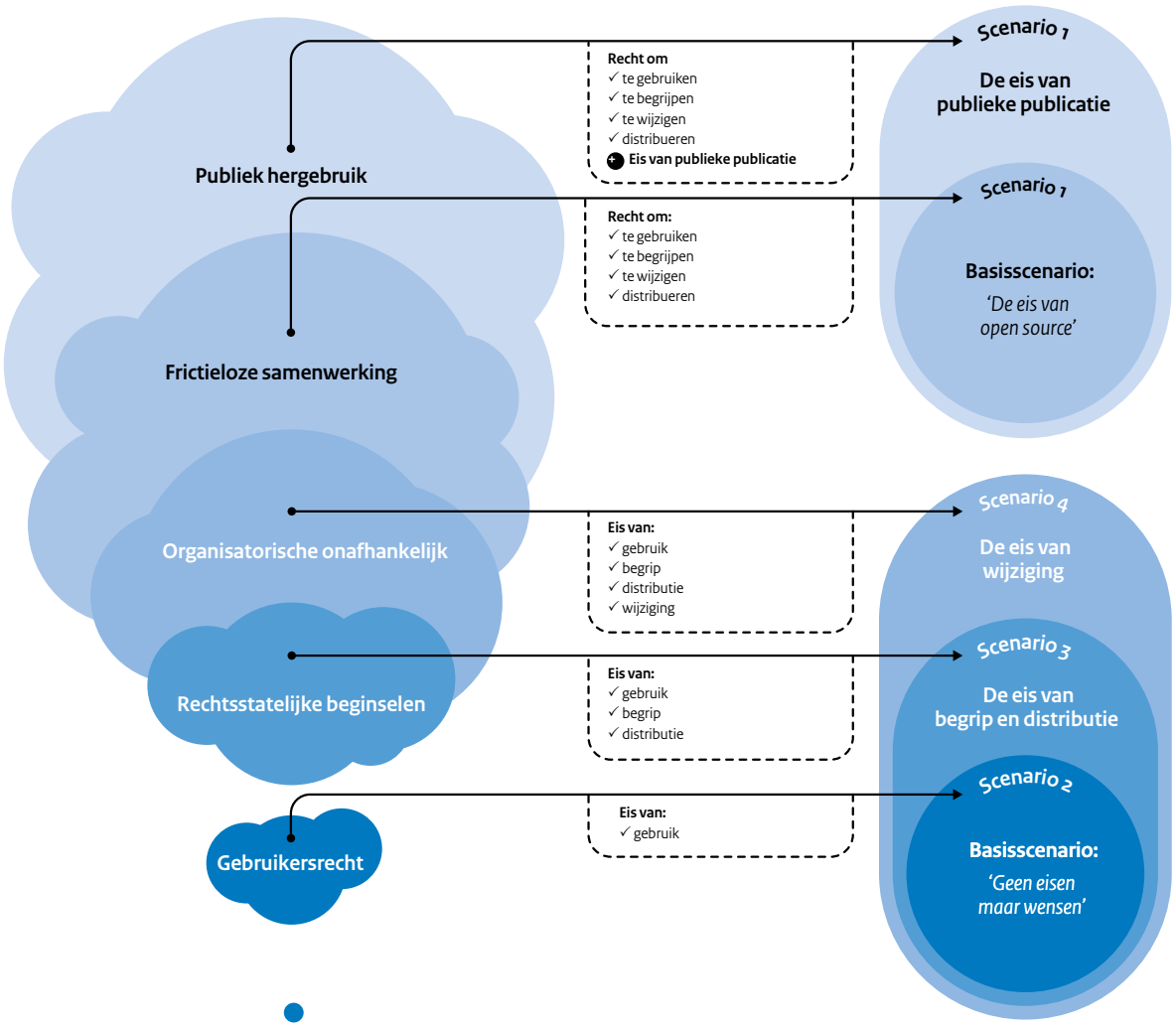
Uiteraard moet blijken uit het marktonderzoek dat de eisen die gesteld worden, ook een haalbare kaart zijn. Wanneer dit niet het geval is, zal dit tot herziening van de strategische visie moeten leiden om uit te werken hoe alsnog tot het gewenste scenario bereikt kan worden.

De negen onderdelen van elk softwareproject

De scenario's bespreken we aan de hand van negen onderdelen die noodzakelijk zijn wil elk softwaresysteem functioneren in de praktijk. Dit document geeft voorbeelden van hoe een project om software aan te besteden kan worden opgedeeld. Ook is in dit hoofdstuk terug te vinden onder welke voorwaarden deze verschillende onderdelen onafhankelijk van elkaar door verschillende leveranciers geleverd kunnen worden. Dit is uiteraard een cruciaal punt voor de aanbesteding: kunnen er verschillende werkpakketten gemaakt worden, die onafhankelijk van elkaar aanbesteed kunnen worden of niet? Het eerste voorbeeld uit hoofdstuk 5 komt in de scenario's terug als illustratie.



Samenvatting van de scenario's



De stappen die vooraf gaan aan de scenario's maken duidelijk welke eisen gesteld worden aan de rechten. Samen met de generieke opdeling van software in onderdelen bepaalt dit hoe de aanbesteding in elkaar *kan* zitten. Wanneer een open source licentie vereist is, kunnen onderdelen als losse werkpakketten aanbesteed worden bij verschillende leveranciers. Dit kan, omdat de voorwaarden er zijn om bedrijven voldoende onafhankelijk van elkaar kunnen laten werken. In het andere extreem wordt de keuze aan de markt gelaten. Het is daarbij *vooraf* niet zeker welke rechten de uitkomst zullen zijn van de aanbesteding. Dat maakt dat verschillende onderdelen moeten worden gebundeld in één grotere aanbesteding. Ook vraagt dit scenario om gedetailleerde afspraken over elke mogelijke wens, omdat de uitkomst kan zijn dat de software (grotendeels) onder private controle staat. De laatste twee scenario's zijn varianten op deze twee basisscenario's.

Scenario 1 - *de eis van een open source licentie*

Dit is het eenvoudigste scenario. Dat komt omdat er al voor de aanbesteding zekerheid is over de rechten die verkregen worden. Daarom kunnen verschillende onderdelen onafhankelijk van elkaar verworven worden.

- De visie beschrijft het belang van de rechten op basis van publieke waarden.
- Uit marktonderzoek blijkt dat er open source software is die kan voldoen.

Kenmerken

- Aanbesteden van de basissoftware is niet noodzakelijk
- Naast de basissoftware worden uitsluitend diensten aanbesteed.
- Naast de basissoftware kunnen de acht elementen van elk softwareproject als losse werkpakketten worden aanbesteed bij verschillende leveranciers.
- Resultaten van diensten worden opgeleverd onder een open source licentie. Dit maakt een uitzondering op de inkoopvoorwaarden soms noodzakelijk.
- Diverse risico's zijn lager, zoals faillissement of de timing van upgrades.
- Er is frictieloze overdracht van software mogelijk tussen alle betrokken partijen.
  Ook is overdracht aan toezichthouders mogelijk en kan extern getoetst worden of en welke kwetsbaarheden in de code zitten en/of hoe snel die opgelost zijn.

Bijzonderheden

1. Het is van belang vooraf een visie te ontwikkelen op de samenwerking. Vaak is er al een werkwijze in het ecosysteem rond de basissoftware, maar moet die expliciet gemaakt worden om een rol te kunnen spelen in de aanbesteding.
2. Het gaat niet uitsluitend om software, bij de oplevering, maar ook om andere assets.
3. Als diensten zijn geformuleerd als overdraagbare taak is het resultaat niet alleen het uitvoeren van zo'n taak, maar ook het documenteren van dat werkproces.
4. Als 3) het geval is, dan kunnen meerdere partijen gelijktijdig werken aan *dezelfde* onderdelen. Dit is met name relevant bij onderhoud en backup & recovery[2].
5. Geen migraties meer, de software kan onderdeel van de organisatie blijven.

Scenario 2 - *geen eisen, maar wensen*

Dit is het meest complexe scenario. Dat komt omdat vooraf niet duidelijk is of verschillende onderdelen van het project onafhankelijk van elkaar uitgevoerd kunnen worden. En dus moeten ze in samenhang worden bekeken. Daarnaast zijn er aanvullende afspraken nodig voor alle gevallen waarin er afhankelijkheid kan zijn van de aanbieder. Dat komt omdat de uitkomst van de aanbesteding kan zijn dat de software die de koper verwerft onder private controle staat.

- De visie beschrijft geen onderbouwing voor de vereiste van aanvullende rechten.
- Marktonderzoek is gericht op financiële onderhandeling over maatwerk(modules).

Kenmerken

- Er is bestaande software die (met aanpassingen) kan voldoen.
- De keuze voor propriëtaire of open source software wordt aan de markt gelaten.
- Omdat vooraf onduidelijk is of andere diensten onafhankelijk kunnen worden uitgevoerd, zullen de verschillende diensten gebundeld met het basissysteem in één grote aanbesteding verworven moeten worden.
- Complex, want diensten en producten moeten in samenhang overwogen worden.
- Het is lastig om diensten als overdraagbare taken te formuleren, omdat de uitkomst kan zijn dat er afhankelijkheden zijn van een product onder private controle.

Bijzonderheden

Aan de voorkeur voor open source software kan invulling gegeven worden door ruime gebruiksrechten hoger te waarderen.

1. Er is extra aandacht nodig voor een eerlijke vergelijking tussen propriëtaire en open source software door "gebruikelijke" verwachtingen expliciet te maken en dus te specificeren en te waarderen, zoals
   - wensen en verwachtingen van onderhoud en (externe) toetsbaarheid
   - risico's, zoals faillissement of keuze in timing van upgrades of migratie
   - de afhankelijkheden van de leverancier bij het oplossen van problemen
2. In plaats van onafhankelijkheid, kan ingezet worden op financiële kortingen in ruil voor het in eigendom geven aan de leverancier van maatwerk
3. Elke mogelijke wens in de toekomst moet zoveel mogelijk nu vastgelegd worden

[2] Het is zelfs mogelijk om de hosting parallel bij meerdere partijen weg te zetten, zelfs op twee locaties onder eenzelfde (ip)adres. Dat vraagt om een architectuurvisie, die input kan zijn bij de aanbesteding.

Scenario 3 - *de eis van inzage en distributie*

Dit scenario is in essentie gelijk aan het vorige scenario. De keuze voor open source of propriëtaire software wordt aan de markt gelaten in dit scenario, wanneer - blijkens het marktonderzoek - er propriëtaire partijen zijn die deze rechten willen bieden. Maatwerkmodules apart aanbesteden is een meer reële mogelijkheid in dit scenario.

- De visie beschrijft de noodzaak van technische transparantie
- Uit marktonderzoek blijkt dat ook propriëtaire aanbieders dit willen bieden

Kenmerken

- Gelijk aan scenario 2

Bijzonderheden

- Gelijk aan scenario 2, behalve:

1. Maatwerkmodules kunnen beter onafhankelijk aanbesteed worden

Scenario 4 - *vereiste van gebruik, begrip, wijziging en distributie*

Dit scenario kan op verschillende manieren worden geïmplementeerd. Dat kan zowel via (eigendom op) maatwerk als via open source software, maar geen van beiden volgen noodzakelijk uit deze eisen.

- De visie beschrijft de noodzaak van alle vier de rechten
- Uit marktonderzoek blijkt geen actief ecosysteem rond open source software

Kenmerken

- Keuze uit meerdere vormen, maar eigendom vereisen maakt de software duur
- Koper wordt (in eerste instantie) beheerder van de software

Bijzonderheden

1. Kan dienen als opmaat naar het scheppen van een (open source) markt
2. Geen eigendom vereisen op maatwerk, vraagt afwijken van inkoopvoorwaarden
3. Sourcen van personeel kan een alternatief zijn voor dit scenario

2. Inleiding op het document

Doel van dit document: waarom nadenken over rechten?

Rechten belangrijk
Rechten spelen een sleutelrol wil de inkoop van ICT van invloed zijn op ambities als innovatie en bedrijvigheid, rekenschap over veiligheid en onafhankelijkheid.

Er is meer van belang dan prijs en functionaliteit bij de inkoop van software. Zo vloeien er belangen voort uit 1) het eigenbelang van de organisatie, zoals leveranciersafhankelijkheid, uit 2) de beginselen van goed bestuur, zoals de eis van democratische verantwoording en 3) de beginselen van behoorlijk bestuur, zoals samenwerking of transparantie. Daarnaast kunnen er 4) specifieke beleidsambities zijn, zoals het stimuleren van (lokale) bedrijvigheid of marktwerking. Al deze waarden hebben een directe link met de rechten die een overheid verwerft bij de inkoop van software en hoe zij dit doet.

Allereerst zullen hieronder zes waarden worden uitgewerkt die een rol kunnen spelen in de strategische visie van een organisatie bij het verwerven van software. Goed zicht op welke waarden een organisatie nastreeft, waar ze vandaan komen en waarom de organisatie ze nastreeft is van belang bij het draagkrachtig[3] motiveren van de eisen die een organisatie stelt en de wensen die ze waardeert in de aanbesteding.

Software, wat kopen we dan?

Wie software verwerft, verwerft een verzameling van rechten, diensten en soms producten. We hebben niet altijd goed zicht op de onderdelen en hun samenhang wanneer ze gebundeld zijn in een businessmodel. Het is daarom belangrijk overzicht te hebben van de onderdelen die noodzakelijk zijn, wil een systeem functioneren in de praktijk. Een visie hierop moet leidend zijn, om te kunnen bezien of een aanbod in de vorm van een bepaald businessmodel aansluit bij de visie van de organisatie.

Framework
Voor het woord 'framework' dat veel gebruikt wordt, kan ook 'oplossing' gelezen worden, het is een bundeling van softwarecomponenten.

We spreken makkelijk over "off the shelf" software of maatwerk alsof het binaire categorieën zijn, terwijl er eerder sprake is van een continuüm. In de praktijk vraagt "off the shelf" software ten minste configuratie en vrijwel altijd maatwerk en maatwerktoevoegingen. Andersom begint maatwerksoftware niet met niets, maar met ten minste met componenten die aan elkaar geschreven worden en meestal met een basissysteem of framework. Daarom is het goed eerst in beeld te hebben welke elementen in vrijwel elk softwareproject nodig zijn, onafhankelijk van hoe deze gebundeld kunnen zijn in een businessmodel.

Aanpak: eisen en wensen aan rechten afleiden uit visie op ambities

Het uitgangspunt van de redenering zou open source software moeten zijn, conform de ambitie van opeenvolgende kabinetten van "open source, tenzij". Echter hier zullen we bij elke stap eerst kijken naar het waarom daarvan aan de hand van specifieke ambities. Daarvoor zijn verschillende redenen. Eén van de redenen daarvoor is om te kunnen beargumenteren wat de achtergrond is van de wensen en eisen. Dat kan bijvoorbeeld relevant zijn als een leverancier een tegenvoorstel doet. Een andere reden is om betere keuzes te kunnen maken in het proces van aanbesteden. Dat is bijvoorbeeld relevant als het gaat om de keuze tussen wensen en eisen of om de weging bij gunningscriteria. Vervolgens zullen we kijken naar het hoe, gegeven de ambities, in verschillende scenario's.

[3] Dit betekent dat je voldoende argumenten hebt om een wens of conclusie te motiveren. Als de argumenten de conclusie helemaal kunnen dragen, dan zijn ze 'draag-krachtig'. In dit geval is dat relevant om in een procedure te kunnen motiveren waarom een alternatief voorstel niet acceptabel is.

3. Rechten en open source software: de essentie

Open source licenties

4 rechten of vrijheden
- Gebruik voor elk doel.
- Begrijpen werking.
- Wijzigen code.
- Verspreiden.

Open source is een verzamelterm voor software onder een set van gestandaardiseerde licenties. Deze licenties geven gebruikers meer dan alleen een gebruiksrecht. Met deze licenties is ook mogelijk om te begrijpen hoe de software werkt, omdat de broncode ingezien kan worden, de mogelijkheid om de software te wijzigen en de mogelijkheid om de software te verspreiden.

Frictieloos samenwerken

Standaard contract
Een open source licentie combineert een ruim gebruiksrecht in een standaard contract met de impliciete acceptatie van dat contract. Daarom werkt het contract als een algemene regel. Dit maakt overdracht in de samenwerking frictieloos.

Omdat deze licenties niet alleen een ruim gebruiksrecht leveren, maar ook gestandaardiseerd zijn én gebruik van de software verbinden aan acceptatie van de licentie, werken deze contracten praktisch gezien als algemene regels[4].

De software kan niet alleen van leverancier naar gebruiker worden overgedragen, maar tevens onderling tussen alle betrokken partijen. Daarmee reduceren ze de transactiekosten, de administratieve lasten en werken als een immateriële infrastructuur voor samenwerking.

Twee categorieën licenties

Twee smaken
Zie paragraaf 1.1 van het playbook open source software op kafkabrigade.nl/playbook

Er zijn in essentie twee soorten open source licenties. Enerzijds de zogenaamde “toegeeflijke” licenties. Deze verlenen de gebruiker alle rechten, ook het recht om de software aan te passen en onder andere voorwaarden te verspreiden. Anderzijds de “beschermende” licenties. Deze licenties verbinden rechten en software onlosmakelijk met elkaar. Wanneer eenmaal het recht is verleend om de software te gebruiken, begrijpen, wijzigen en distribueren, dan blijven die rechten verbonden aan de software wanneer die wordt overgedragen. In het eerste geval kan de software weer onder private controle gebracht worden in het tweede geval blijft de software altijd in dezelfde mate toegankelijk voor alle partijen aan wie de software wordt overgedragen.

[4] Zie ook hoofdstuk 8 van Widlak, 2021a.

Geen vereiste om wijzigingen vrij te geven of plicht tot openbare publicatie

Doorgaans wordt verondersteld dat open source licenties vereisen om wijzigingen vrij te geven of de software openbaar te publiceren[5]. Dat is geen vereiste[6]. De eis is slechts om de broncode mee te leveren indien je de software verspreidt, slechts aan de ontvanger(s) en alleen wanneer dit een beschermende licenties is. Wie binnen de eigen organisatie open source software inzet is tot niets verplicht[7]. Toch is het niet vreemd dat deze associatie gemaakt wordt, omdat dit wel de gangbare praktijk is. De frictieloze overdracht die open source licenties mogelijk maken, maken daarmee ook een frictieloze open samenwerking mogelijk en aantrekkelijk. Echter andersom staat een open source licentie een besloten werkwijze niet in de weg, zoals die - voor specifieke onderdelen - misschien nodig kan zijn voor een vertrouwelijke werkwijze.

De essentie

De essentie, nodig om dit document te begrijpen, is dus dat het verwerven van software het verwerven van rechten impliceert. Er zijn vier belangrijke rechten, die verbonden zijn met de aard van wat software is:

1. het recht om de software te gebruiken
2. het recht om de software te begrijpen en dus toegang te hebben tot de broncode
3. het recht om de software te wijzigen, waarvoor 2) een voorwaarde is
4. het recht om de software (gewijzigd) te distribueren

Afhankelijkheid
Afhankelijkheid ontstaat als de verwerver software gebruikt, die niet zomaar is in te wisselen en die gelijktijdig onder controle blijft staan van de aanbieder.

Het verwerven van rechten gaat over controle of vrijheid. Wanneer bepaalde rechten in bezit zijn van een private partij en niet in het bezit van de verwerver, is er op dat punt – onbegrensd in de tijd – sprake van controle door deze partij, ook al is de software in gebruik bij de verwerver. Wanneer deze rechten niet verworven worden – bijvoorbeeld om in plaats daarvan een korting te bedingen – wordt het veel belangrijker sluitende afspraken te maken. Sluitende afspraken zijn in zo'n geval nodig, omdat met verwerving afhankelijkheid ontstaat van de aanbieder. Wie later aanvullende wensen heeft, zal toestemming moeten vragen aan de aanbieder. Uiteraard zijn er ook andere mechanismen waardoor afhankelijkheden kunnen ontstaan, maar dit document richt zich op de *voorwaarden* om die afhankelijkheden weg te nemen of te reduceren: de rechten.

[5] We zien die veronderstelling bijvoorbeeld terug in de Kamerbrief inzake het vrijgeven van de broncode van overheidssoftware. Deze brief spreekt over "de benodigde vertrouwelijke werkwijze van de overheid, denk aan opsporing en toezicht" als mogelijke grond om geen open source software te gebruiken. Het is echter mogelijk om software te ontwikkelen en deze onder een beschermende open source licentie over te dragen aan mede-overheden, zonder deze openbaar te publiceren.

[6] Een bespreking van dit vraagstuk is hier te vinden: https://platformoverheid.nl/artikel/geheime-open-source-software/ of https://ibestuur.nl/podium/geheime-open-source-ftware. Zie ook bijvoorbeeld https://www.gnu.org/licenses/gpl-faq.html#DoesTheGPLRequireAvailabilityToPublic, of Arnoud Engelfriet hierover op: https://www.security.nl/posting/24818. Zie ook voetnoot 22. En met dank aan Maurice Hendriks die mij hierop wees.

[7] Uiteraard heeft elke organisatie er wel *belang* bij om dit te doen, al was het maar om niet bij elke versie de eigen aanpassingen opnieuw te moeten doorvoeren.

4. Waarom aanbesteden voor publieke waarden?

De inkoop van software die overheden gebruiken voor hun primaire proces, dus waaraan zij individuele besluitvorming delegeren, gegevensuitwisseling of data-analyse, heeft direct invloed op democratische en rechtsstatelijke waarden. Daarnaast kan de inkoop ook direct invloed hebben op andere beleidsdoelen die de overheid namens ons nastreeft, zoals digitale veiligheid, gezonde marktwerking of lokale bedrijvigheid. En zelfs de invloed van de inkoop van ICT op "slechts" de bedrijfsvoering kan belangrijk zijn voor bestuurlijke waarden als continuïteit en wendbaarheid.

Net zoals de inkoop van producten invloed heeft op het milieu en de inkoop dus altijd ook een beleidshandeling is conform of contra het milieubeleid, is de inkoop van software van invloed op allerlei beleidsdoelen. Dat is onafhankelijk van de vraag of de inkoper hierin bewust een keuze maakt of niet. Wederom: net als bij duurzaam inkopen. De inkoop van software heeft invloed op de autonomie van een organisatie, op haar vermogen zich te verantwoorden of samen te werken en vele andere zaken. Het gaat zowel om belangen van de organisatie zelf, zoals continuïteit en wendbaarheid, maar ook om belangen van de samenleving als geheel[8], zoals verantwoording.

Waarden en voorwaarden

Alles waarvoor we moeite willen doen, is een waarde. In een aanbesteding kan dat de prijs zijn of een specifieke functionaliteit, maar ook meer abstracte zaken. Continuïteit is een meer abstracte waarde, die van groot belang is in het primaire proces.

Sommige waarden vinden we tevens een morele plicht. We noemen zulke waarden beginselen[9]. Verantwoording is zo'n beginsel. De Nederlandse code voor goed openbaar bestuur[10] noemt als één van de zeven beginselen van deugdelijk overheidsbestuur dat het bestuur bereid is zich regelmatig en ruimhartig te verantwoorden jegens de omgeving.
Een beginsel als verantwoording vraagt om voorwaarden. De bereidheid om zich te verantwoorden, vraagt om de mogelijkheid zich te verantwoorden. In een context waarin overheidshandelen is gedelegeerd aan software, zoals bij automatische besluitvorming, gegevensuitwisseling of data-analyse, bepalen de rechten op deze software de mate waarin de overheid zich kan verantwoorden. Hoe de rechten op software worden verdeeld door het proces van aanbesteding bepaalt dus de mogelijkheden tot verantwoording. Waarden komen zo bij de verwerving in een verdelingscontext te staan. We noemen waarden dan belangen[11].

Verantwoording is een publiek belang. Veel beginselen zijn tevens wettelijke verplichtingen[12]. Omdat het publieke belangen zijn, zijn ze in de wet vastgelegd. Met andere woorden: verantwoording kan het verwerven van rechten naast het gebruiksrecht vereisen vanuit strategische, praktische én juridische overwegingen.

[8] Zie het playbook open source software voor een meer uitgebreide bespreking.
[9] Van der Heijden, 2001:3.
[10] https://www.rijksoverheid.nl/onderwerpen/kwaliteit-en-integriteit-overheidsinstanties/gedragscode-openbaar-bestuur.
[11] Widlak, 2021a:34. Belangen zijn waarden in een verdelingscontext.
[12] Zoals de beginselen van behoorlijk bestuur, die zijn opgenomen in de Algemene wet bestuursrecht.

Bronnen van publiek waarden

Er zijn diverse documenten die beginselen, zoals verantwoording, benoemen en meer of minder concreet uitwerken. In veel gevallen vragen deze beginselen nadere uitwerking om hieruit conclusies te kunnen trekken met betrekking tot aanbesteding. In noem hier enkele voorbeelden en licht daar enkele aangrijpingspunten voor de aanbesteding uit.

1. **De code goed openbaar bestuur uit 2009**
   De code goed openbaar bestuur[13] benoemt zeven beginselen: openheid en integriteit, participatie, behoorlijke contacten met burgers, doelgerichtheid en doelmatigheid, legitimiteit, lerend en zelfreinigend vermogen en verantwoording. “Openheid”, zo benoemt de code, “betekent in ieder geval dat het bestuur open is over procedures en besluiten. Ook maakt het bestuur relevante informatie toegankelijk.”
2. **De code goed digitaal openbaar bestuur uit april 2021[14]**
   De code goed digitaal openbaar bestuur benoemt drie fundamenten – democratie, rechtsstaat en bestuurskracht. Deze fundamenten vult zij in met principes die zij toepast in de context. Zo vraagt de rechtsstaat om procedurele rechtvaardigheid en dat impliceert onder meer uitlegbaarheid van het digitale proces. En zo vraagt bestuurskracht om verantwoordelijkheid en bestuurskwaliteit en dat impliceert wendbaarheid van haar digitale infrastructuur. De code goed digitaal bestuur benoemt bij de “voorbeelden van organisatorische acties om deze waarden te borgen” expliciet de rol van open source software bij aanbestedingen. Dit om te zorgen dat de organisatie niet afhankelijk wordt van een specifieke leverancier[15].
3. **De beginselen van behoorlijk bestuur, zoals opgenomen in de Algemene wet bestuursrecht[16], de Europese code voor goed administratief gedrag[17] en het recht op goed bestuur uit artikel 41 van het EU-Handvest van de grondrechten[18].**
   In de kern beschrijven deze beginselen wat we rechtvaardig vinden in relaties met grote machtsverschillen, zoals die tussen overheid en burger[19]. Dit zijn tevens eisen die we stellen aan organisaties wanneer ze besluiten nemen. Een organisatie is verantwoordelijk voor zo’n besluit. Dat betekent onder meer dat die beslissing, en het proces dat ertoe heeft geleid, navolgbaar moet zijn. Er is geen algemene en onverkorte plicht uit af te leiden tot technische transparantie, maar afhankelijk van de context kan die plicht er wel zijn[20]. En daarnaast is er altijd het bestuurlijke risico dat een rechterlijk uitspraak dwingt tot technische transparantie aan procespartijen.

De bovenstaande beginselen, zal ik verder aanduiden met rechtsstatelijke waarden. Het zijn beginselen die samenhangen met democratie, rechtsstaat en goed bestuur. Daarnaast zijn er waarden die samenhangen met beleidsambities en waarden die samenhangen met organisatorische belangen. Vaak is er een overlap. Ze zijn eerder uitgewerkt in het Playbook Publieke Software[21] en worden hieronder geciteerd met aanpassingen:

Continuïteit

Wanneer de rechten beschikbaar zijn om software te begrijpen en te wijzigen, betekent dit in praktische zin dat je die software kunt beschouwen als permanent onderdeel van je eigen organisatie. Je kunt meerdere aanbieders organiseren rond de software die je als organisatie gebruikt. Dat is belangrijk, omdat software in de praktijk als een infrastructuur is voor organisaties. Via software werken mensen samen en organisaties delegeren beslissingen aan software.

---

[13] https://www.rijksoverheid.nl/onderwerpen/kwaliteit-en-integriteit-overheidsinstanties/gedragscode-openbaar-bestuur.

[14] https://www.rijksoverheid.nl/documenten/rapporten/2021/04/30/code-goed-digitaal-openbaar-bestuur.

[15] Op pagina 39 en 40.

[16] https://wetten.overheid.nl/BWBR0005537/2021-11-01#Hoofdstuk2_Afdeling2.1_Artikel2:1.

[17] https://www.ombudsman.europa.eu/nl/publication/nl/3510.

[18] https://eur-lex.europa.eu/legal-content/NL/TXT/?uri=CELEX:12012P/TXT.

[19] Widlak, 2021a:298, 302, 348.

[20] Op de vraag wanneer technische transparantie wettelijk verplicht is, is alleen een genuanceerd antwoord mogelijk. Technische transparantie impliceert het kunnen begrijpen van de software en dus toegang tot de broncode. Ook de vraag voor wie technische transparantie vereist is, kan alleen genuanceerd beantwoord worden. Het is duidelijk dat de beslisregels - in de zin van in businessrules vertaalde wetgeving - onder de Wet openbaarheid van bestuur vallen (https://www.tweedekamer.nl/kamerstukken/kamervragen/detail?id=2018Z02454&did=2018D22110). Daarnaast is er de software die de procedure (2) regeert, waarin de toepassing van beslisregels een plaats heeft. Ook die procedure moet navolgbaar zijn, daarover bestaat geen twijfel. Die software kan in juridische zin ook een beslisregel zijn, wanneer de wet eisen stelt aan het proces. Echter uit het huidige bestuursrecht kan nog geen algemene en onverkorte verplichting tot technische transparantie kan worden afgeleid. Of technische transparantie vereist is, is afhankelijk van de rol die de normen – en daarmee de software – vervullen in het besluitvormingsproces. Soms is er een verplichting tot algemene bekendmaking, soms kan er een verplichting zijn tot bekendmaking aan procespartijen en soms is het mogelijk dat openbaarmaking achterwege blijft, vanwege gewichtige redenen, wanneer er geen sprake is van een beleidsregel (Wolswinkel, 2020:40-43). Johan Wolswinkel beschrijft in zijn oratie dat de vraag naar een wettelijke verplichting afhankelijk is van de specifieke context.

[21] http://kafkabrigade.nl/playbook of https://www.rijksoverheid.nl/documenten/brochures/2021/01/05/playbook-publieke-software-aanbesteden.

In het ideale geval behoud je daarom die software en bouw je daarop voort, ook wanneer je van aanbieder wisselt. Dat zorgt voor continuïteit en kan in specifieke gevallen potentieel moeilijke en tijdrovende migratietrajecten voorkomen.

Leveranciersonafhankelijkheid

De mogelijkheid om van aanbieder te wisselen, zonder van software te wisselen, maakt je minder afhankelijk van een specifieke leverancier. Behalve wisselen van aanbieder, is het ook mogelijk om meerdere aanbieders tegelijk aan dezelfde software te laten werken. Dat kan in eenzelfde rol, waarbij meerdere aanbieders gelijktijdig werken aan uitbreidingen, verbeteringen, het wegwerken van fouten of in de werkzaamheden die nodig zijn om software beschikbaar te maken als een dienst via de cloud. En dat kan ook in complementaire rollen, waarbij de ene leverancier nieuwe onderdelen ontwikkelt, een andere aanbieder die integreert en een derde aanbieder die onderhoudt. Omdat je de ontwikkeling op zo'n wijze kunt organiseren, organiseer je tevens dat meerdere partijen inhoudelijke kennis hebben van de software. Je organiseert dat aanbieders elkaars documentatie testen en meer zaken die het reëel maken dat verschillende aanbieders elkaars werk kunnen overnemen. Dit betekent ook dat juridische waarborgen, zoals Escrow[22], niet nodig zijn. Zulke waarborgen zijn dan georganiseerd - opgenomen in de werkwijze - en daarmee niet juridisch, maar praktisch geborgd.

Leveranciersonafhankelijkheid en continuïteit kunnen zo dan ook zelfversterkend werken.

Marktwerking

Open source schept de mogelijkheid om kennis te nemen van de werking van de software. Naarmate meer aanbieders die kennis ook daadwerkelijk hebben - omdat een bredere groep aanbieders betrokken is bij de software - is het beter mogelijk om een marktordening te organiseren die daadwerkelijk tot marktwerking leidt. Naarmate overheden meer code actief delen, is het aantrekkelijker voor bedrijven om te investeren in kennis van die code. Er zit veel overlap in de code van overheidsorganisaties en het werk daaraan. Het kan een doel op zich zijn om code te kunnen delen, bijvoorbeeld omdat dit een kostenbesparing kan opleveren. Maar breed gebruik van dezelfde code maakt die ook interessanter voor bedrijven. Zij kunnen met die kennis op meerdere plekken aan de slag[23]. Ook dat is een stap op weg naar een marktordening die daadwerkelijk tot marktwerking leidt. Uiteraard vergt dit een strategie[24] waarin actief beheer van de software is afgestemd op het actief organiseren van een bredere groep van aanbieders.

Samenwerking en ketensamenwerking

Ongehinderd samenwerken over organisatiegrenzen heen is steeds belangrijker en tevens de kern van wat open source software mogelijk maakt. Samenwerking tussen organisaties is vandaag ook altijd een kwestie van samenwerking tussen softwaresystemen ofwel interoperabiliteit[25].

Technische transparantie draagt daaraan bij. Hoe beter je weet hoe iets werkt, hoe makkelijker het is om daarop aan te sluiten. Dat geldt voor overheden onderling, maar dit maakt het natuurlijk ook voor bedrijven makkelijker om aan te sluiten op systemen en te anticiperen. Wanneer overheden onderling samenwerken en één van hen gebruikt software die onder private controle staat kan dit onderling aansluiten lastig maken. Dit schept ruimte voor leveranciers om private belangen boven het belang van samenwerking of het belang van de keten te stellen. Zij kunnen dan samenwerking op het niveau van software onmogelijk maken. Dit is inefficiënt vanuit overheidsperspectief, omdat meerdere partijen dan in hun eigen cocon aan hetzelfde probleem moeten werken, maar uiteraard voordelig vanuit het perspectief van een aanbieder.

Innovatie en bedrijvigheid

Open source licenties dwingen niet[26] tot publieke publicatie van de broncode die in gebruik is. Echter het is niet slechts voor onderwijs en wetenschappelijk onderzoek van groot belang dat de broncode van software gepubliceerd

[22] Een broncode-escrow is een juridische overeenkomst. De leverancier van software deponeert de broncode bij een derde partij. Deze derde partij geeft de broncode vrij in bijzondere gevallen, zoals faillissement.

[23] De investering in het kennisnemen van de interne werking van de software verschuift hiermee ook deels van de kopende partij (de overheid), naar de marktpartijen.

[24] In beleidstermen: de regierol en governance zijn essentieel.

[25] In beleidsstukken staat op dit punt vaak dat open source software geen voorwaarde is voor interoperabiliteit en open standaarden wel. Dat is niet helemaal juist. Maar belangrijker is dat dit beeld doet geen recht aan de praktijk. Open standaarden en open source software zijn verschillende werelden. Open standaarden hebben hun oorsprong bij grote bedrijven, ontwikkelen zich langzaam en hebben een verre tijdshorizon. Ze ontstaan indien meerdere bedrijven voldoende belang hebben om na jarenlange onderhandelingen een specificatie te implementeren. Open source software ontwikkelt zich sneller en heeft een meer pragmatische insteek. Open source software gebruikt open standaarden als ze er zijn en anders is een onofficiële microstandaard snel ontwikkeld. Open standaarden en open source software versterken elkaar in termen van interoperabiliteit, mede door deze verschillen. Een open standaard is er niet altijd. En open standaarden zijn - soms moedwillig - niet altijd juist geïmplementeerd. Open source software maakt dergelijke problemen op een pragmatische manier en op een redelijke termijn oplosbaar, omdat de implementatie transparant is. Dat maakt tevens de ontwikkeling van een open standaard en de toetsing of de werking juist is makkelijker.

[26] Een uitzonderingssituatie is wanneer software onder de Affero General Public Licencse (AGPL) ingezet wordt voor een publieke online dienst.

beschikbaar is. Inmiddels rust de IT industrie in elke sector op open source software[27]. De enorme snelheid waarmee vandaag een idee in een concrete toepassing kan worden gegoten, kan alleen bij de gratie van de brede beschikbaarheid van open source componenten. Of het nu gaat om databases, (front-end) frameworks of modules, de mogelijkheid om in relatief korte tijd een complete toepassing te realiseren zou voor slechts een fractie van de ontwikkelaars open staan zonder open source software. De investeringen in tijd en geld zouden anders simpelweg te hoog zijn. In de praktijk zien we dan ook dat een groot en steeds groter deel van de commerciële software open source componenten bevat. Volgens een jaarlijkse audit[28] van inmiddels meer dan 1500 commerciële softwarepakketten bevatte elk pakket open source componenten en gemiddeld 75% van de code bestond uit open source software. Commerciële propriëtaire software is in de praktijk dus voor het grootste deel open source software. Starten als bedrijf zonder een basis of aanvulling in open source software is nauwelijks meer reëel. Maar met open source software kan een startend bedrijf meeliften op miljoenen regels code die zich al in de praktijk hebben bewezen[29].

Veiligheid en verantwoording

Openheid en veiligheid in de betekenis van cybersecurity lijken voor veel mensen intuïtief tegengesteld.

Het intuïtieve argument is dat een fout die onbekend is, niet misbruikt kan worden. Het tegenargument is dat het beter is om in alle openheid een systeem te bedenken waarvan je kunt beredeneren dat het veilig is. Cryptografie[30] is het ultieme voorbeeld daarvan. Je demonstreert de veiligheid en juist de consensus onder kritische kenners geeft je vertrouwen. Empirisch[31] blijken veelgebruikte open en gesloten systemen[32] elkaar weinig te ontlopen in termen van de ernst en aantallen kwetsbaarheden. De onderzochte open systemen publiceren die kwetsbaarheden wel sneller en lossen ze veel sneller op. Het open source concept “vele ogen maken fouten schaars” gaat in die zin op[33].

In de praktijk echter gebruiken organisaties niet alleen grote en goed onderhouden systemen. Dat geldt zowel voor open als gesloten software. En wanneer organisaties zelf functionaliteit laten ontwikkelen, is die niet altijd onderdeel van een brede discussie. Deze praktijk is waarschijnlijk veel belangrijker voor de veiligheid.

Wie software gebruikt, wil daarom zeker stellen dat iemand die software onderhoudt en kwetsbaarheden oplost, of dit nu een (impliciet) onderdeel is van de licentiekosten bij gesloten software - waarbij je de leverancier moet vertrouwen - of van een (expliciet) onderhoudscontract bij open source software, waarbij je dit zelf kunt controleren en publiek kunt verantwoorden[34].

Samenvatting

Al deze waarden hebben een relatie met rechten op de software. Dat betekent niet dat het verwerven van bepaalde rechten voldoende voorwaarde zijn om deze waarden te realiseren. Zo is het mogelijk een maatwerk module voor een gesloten systeem onder een open source licentie te laten opleveren. Daarmee is geen leveranciersonafhankelijkheid verkregen, omdat er daarnaast ook een technische afhankelijkheid is van propriëtaire software. Er zullen daarom ook afspraken nodig zijn met de leverancier van de basissoftware over de stabiliteit van de interface of API, zodat wijzigingen in de basissoftware niet tot problemen leiden. Ook betekent dit niet dat het verwerven van rechten altijd een noodzakelijke voorwaarde is voor bepaalde waarden. Zo kan verantwoording ook worden zekergesteld zonder publieke technische transparantie. Denk hierbij aan verantwoordingssystemen, zoals in de accountancy. Wel kunnen we concluderen dat:

[27] Het gaat om 100% in marketing en tech, 98% in de gezondheidszorg, 97% in de financiële dienstverlening en 92% in de retail en e-commerce van de software die rust op open source software. OSSRA, 2021.

[28] Black Duck by Synopsis brengt jaarlijks het OSSRA rapport uit, een analyse van veiligheidsaspecten van open source software. De editie van 2021 moet worden aangevraagd (https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html).

[29] Zie ook: Edo Plantinga, Johan Groenen, Hilbrand Bouwkamp, Ad Gerrits, Winst op (tien) punten https://ibestuur.nl/nieuws/winst-op-tien-punten.

[30] Dit is het versleutelen van gegevens, zodat je onbespied kunt internetbankieren bijvoorbeeld.

[31] Schryen, 2009, 2010, 2011.

[32] Er is een open en gesloten browser, email client, webserver, officepakket, besturingssysteem en database management systeem vergeleken.

[33] Wet van Linus: “Given enough eyeballs, all bugs are shallow” (Raymond, 1999). In reactie hierop hebben andere auteurs deze ‘wet van Linus’ enigszins genuanceerd door zich over de samenstelling van de gebruikers het volgende af te vragen: “is it a matter of enough eyeballs, or the right eyeballs?” Uit de vraagstelling lijkt te volgen dat de ontwikkelingsmethode alleen succesvol kan zijn indien er voldoende gebruikers met relevante programmeerkennis zijn. De andere gebruikers moeten er daarnaast in ieder geval positief tegenover staan om mee te helpen aan de kwaliteitsverbetering door mogelijke fouten te melden. Zie ook Playbook Publieke Software Aanbesteden v2.oen https://ibestuur.nl/podium/is-open-source-goed-voor-de-veiligheid.

[34] Belangrijk is dat je weet en kunt weten of een kwetsbaarheid aanwezig is in de software of in een (open source) afhankelijkheid. Wanneer je dat weet, kun je maatregelen treffen, desnoods door software tijdelijk af te koppelen van internet of uit te zetten.

1. Beginselen vanuit een technisch perspectief – zoals technische transparantie – zijn nog geen onderdeel van het bestuursrecht en bestaande beginselen zijn nog niet door de rechter uitgelegd op een wijze dat ze technische transparantie volledig afdekken[35].
2. Tegelijk is het verwerven van rechten meestal een voorwaarde voor het realiseren van bovengenoemde waarden en er is altijd wel een waarde waarvoor deze rechten een noodzakelijke voorwaarde zijn.
3. Het onderbrengen van rechten in een standaardcontract met ruime rechten dat impliciet geaccepteerd wordt - een open source licentie dus - vereenvoudigt de realisatie van bepaalde waarden sterk. Dat komt met name omdat de overdracht van code veel eenvoudiger is.
4. Hoe eisen of wensen aan de rechten in een aanbesteding vorm krijgen hangt daarom allereerst samen met een strategische visie. Daarin staat op de eerste plaats welke waarden de organisatie wil realiseren, hoe ze die wil realiseren en daarmee waarom die rechten nodig zijn.

Conclusies

Hieronder zal ik – om de complexiteit te reduceren – het bovenstaande samenvatten tot vier sets van waarden die een organisatie kan nastreven:

Waardenset 1 “rechtsstatelijke beginselen”

Hierbij beperk ik me tot het volledig kunnen navolgen van de procedure van besluitvorming, beslisregels, bestuurlijk handelen en de verzameling van feiten. Uiteindelijk kan elk onderdeel van een systeem dat gebruikt wordt voor automatische besluitvorming, gegevensuitwisseling of data-analyse directe impact hebben op de rechten en plichten van de individuele burger[36]. Het kunnen bestuderen van de code[37] is een voorwaarde voor transparantie over de procedure.

Uiteraard is de eis om de code te mogen begrijpen in deze context niet slechts het recht om de code te mogen begrijpen als koper. Het is juist van belang dat ook anderen de code mogen begrijpen, zoals een toezichthouder, een rechter, een procespartij, een burger of het publiek als geheel. Het verwerven van het recht om te begrijpen, doe je als overheid juist ten behoeve van anderen. Het recht om te begrijpen staat in de context van aanbesteden dus nooit op zichzelf. De koper moet de code ook mogen distribueren. In de praktijk zal de organisatie ook het recht moeten verwerven om zo’n derde de mogelijkheid te geven de code uit te voeren, ten minste voor onderzoeksdoeleinden. De vrijheid van distribueren is dus distribueren zoals we dat kennen van open source licenties: de ontvanger verkrijgt dezelfde rechten en mag de code niet alleen inzien, maar ook gebruiken.

Waardenset 2 “organisatorische onafhankelijkheid” - omvat tevens waardenset 1

Hierbij gaat het om de voorwaarden om meerdere leveranciers aan dezelfde code te kunnen laten werken. Het gaat dan om het verwerven van alle vier de rechten die in de aard van software liggen. De keuzes die een organisatie maakt, gegeven deze ambitie, zijn in principe dezelfde als bij een keuzes voor continuïteit en marktwerking.

[35] Wolswinkel, 2020.
[36] Widlak, 2021a:161-209 voor de overkoepelende analyse, voorbeelden op diverse andere pagina’s.
[37] Hoe die ook verdeeld is over configuratie (inclusief business-rules) of softwarecode.

Waardenset 3 “frictieloze samenwerking” - omvat tevens waardenset 1 en 2

Leveranciersonafhankelijkheid - in de zin van waardenset 2 hierboven - maakt het verwerven van de vier rechten tot een noodzakelijke voorwaarde, want je kunt alleen meerdere partijen aan dezelfde software laten werken als je die rechten ook hebt en kunt overdragen. Maar die eis maakt naar de letter niet een open source licentie noodzakelijk. Het bijzondere van open source licenties is dat ze overdracht van de code frictieloos maken. Partijen kunnen gezamenlijk aan de code werken en gaan *daarmee* de licentieovereenkomst impliciet aan. Wanneer frictieloze samenwerking tussen organisaties de ambitie is, vereist dit naast de rechten ook een manier om tot een frictieloze overdracht van de code te kunnen komen. Een open source licentie is de nu enige bekende vorm van een gestandaardiseerd contract met voldoende ruime gebruiksrechten dat dit mogelijk maakt. De keuzes die een organisatie maakt, gegeven deze ambitie, is dezelfde als voor de ambities zoals beschreven onder veiligheid en verantwoording. Want ook daarin speelt frictieloze overdracht een centrale rol. Kan een andere partij zeker stellen dat updates verwerkt zijn? Kan een andere partij zeker stellen dat de software onderhouden wordt en kwetsbaarheden verholpen zijn, al dan niet door andere partijen? Frictieloze overdracht is ook van belang voor elk ander systeem van verantwoording met externe partijen gericht op andere waarden dan veiligheid.
En tenslotte heeft de log4j kwetsbaarheid eens temeer zichtbaar gemaakt dat (kunnen) *weten* dat iets onveilig is, de veiligheid ten goede komt. Want dan is software eventueel uit te schakelen of af te koppelen. Dit kan niet bij onbekende kwetsbaarheden of software waarvan je niet weet of (open source) componenten gebruikt worden.

Waardenset 4 “publiek hergebruik” - omvat tevens waardenset 1, 2 en 3

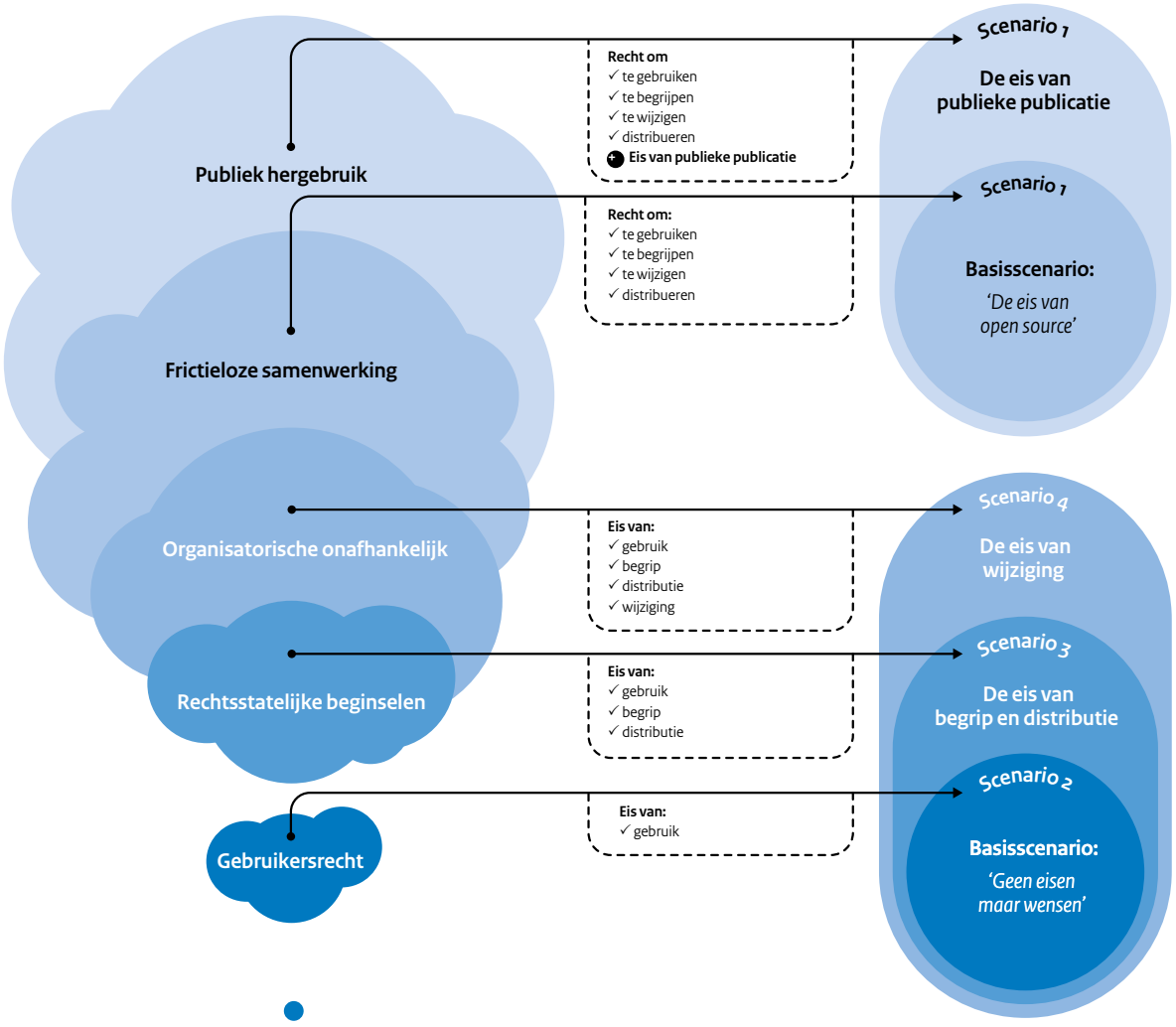
Publieke beschikbaarheid van de code kan een waarde op zichzelf zijn. Dat kan een waarde op zichzelf zijn in een rechtsstatelijke context, omdat transparantie een waarde op zichzelf is en omdat dit een middel kan zijn om vertrouwen te scheppen[38]. Wil de software die ontwikkeld is een basis kunnen zijn voor nieuwe economische activiteit- voor innovatie[39] en bedrijvigheid - op een gelijk speelveld[40], dan vereist dit ook publieke beschikbaarheid. Dan kan elke onderneming hierop voortbouwen. Denk hierbij in praktische zin aan publicatie op een platform als Github.
Hierboven zijn dus vier sets van waarden samengesteld naar aanleiding van de bronnen van publieke waarden uit hoofdstuk 4. De eerste set vraagt niet om open source software, maar wel om het recht de software te kunnen inzien en begrijpen. De tweede set vraagt om dezelfde rechten als open source software, maar vereist dat naar de letter niet. Dit zou ook kunnen via overdracht van eigendom of de combinatie van eigendom op maatwerk en licensering van afhankelijkheden. De derde set vraagt wel om een gestandaardiseerd contract waaronder alle partijen vrijelijk kunnen samenwerken, zoals een open source licentie dat mogelijk maakt. De vierde set vraagt om méér dan open source alleen, namelijk tevens om openbare publicatie.

Om al deze mogelijkheden op een toegankelijke manier te kunnen beschouwen in hoofdstuk 7, bespreken we die aan de hand van vier scenario’s. Enerzijds zullen we de eis van een open source licentie tegenover de situatie zetten waarin geen andere eis gesteld wordt dan het gebruiksrecht.
Dit zijn de basisscenario’s. We zullen dan zien dat waardenset 1 “rechtsstatelijke beginselen” een variant is op het basisscenario waarin alleen een gebruiksrecht wordt geëist. Dit is scenario 3. Waardenset 4 “publiek hergebruik” blijkt een variant op het scenario waarin open source software wordt vereist. Omdat dit zo weinig verandert aan scenario 1, zullen we die tegelijk bespreken. Tenslotte blijkt waardenset 2 “organisatorische onafhankelijkheid” naar de letter een variant op basisscenario 2. Dit is scenario 4. Maar tevens zal blijken dat het makkelijker is hier scenario 1 te volgen.

[38] Dit was hét motief van VWS bij de CoronaMelder. VWS toont zelfs aan exact welke code op je telefoon komt te staan.

[39] Op de relatie met innovatie zijn twee visies mogelijk. De ene is dat het wenselijk is dat intellectueel eigendom opnieuw geclaimd kan worden door een bedrijf, zodat deze met een relatief kleine toevoeging rendement kan halen op het intellectueel kapitaal van het geheel. Apple is een typisch voorbeeld van dit model. De andere visie is dat het wenselijk is dat bedrijven juist een marge kunnen maken op diensten die zij in combinatie kunnen aanbieden. Een typisch voorbeeld van dit model is Red Hat. Verschil in visie leidt hier tot een verschil in licentie.
Zie hiervoor de licentiewijzer die zal verschijnen (of is verschenen) op https://opensource.pleio.nl/.

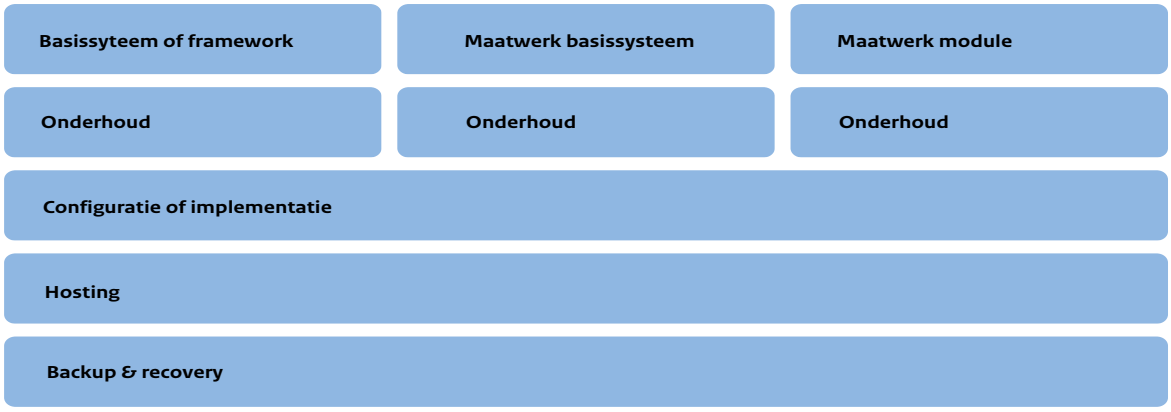
[40] Wanneer software niet voor elk bedrijf beschikbaar is en wel publiek gefinancierd dan schendt dit een belangrijk rechtvaardigheidsprincipe. Het is bovendien niet effectief in het licht van het stimuleren van innovatie en bedrijvigheid. Daarnaast kan - ook als gekozen wordt voor publieke beschikbaarheid - er nog altijd voor gekozen worden om het mogelijk te maken dat bedrijven zich de code toe-eigenen wanneer zij daarop voortbouwen, door te kiezen voor een open source licentie die dit toestaat. Dit zijn de zogenaamde toegeeflijke licenties, zoals de BSD-licentie.



Voordat we iets zullen zeggen over de consequenties voor en strategische overwegingen in een aanbesteding van deze eisen is het goed om eerst iets te zeggen over software.

5. Software, wat kopen we dan?

Wie software verwerft, verwerft altijd een verzameling van diensten en producten. Soms zien we die onderdelen niet, zoals bij software as a service. Soms kopen we enkele onderdelen uit die verzameling. Maar het is goed om enig zicht te hebben op de onderdelen die noodzakelijk zijn voor elk systeem dat functioneert in de praktijk.



Software wordt wel gecategoriseerd naar de businessmodellen die bedrijven gebruiken. Zo spreken we over *"off the shelf"* om aan te duiden dat het een kant en klaar pakket betreft dat na installatie direct gebruikt kan worden. Al blijkt de praktijk vaak anders, het idee is dat alleen een basissysteem gekocht hoeft te worden en dat eventuele andere onderdelen uit de figuur hierboven van een verwaarloosbare betekenis zijn. Of we spreken over "maatwerk" om aan te duiden dat de eisen van de klant zo specifiek zijn, dat oplevering in elk geval heel veel arbeid vereist, waarbij de indruk kan ontstaan dat er met niets begonnen wordt.

In werkelijkheid is dit onderscheid niet zo binair. Dat geldt zeker voor de context hier, waar we het niet hebben over tekstverwerkers, maar over systemen in het primaire proces die zorgen voor automatische besluitvorming, gegevensuitwisseling of data-analyse. Elk *"off the shelf"* product vereist configuratie, vrijwel altijd "implementatie" of aanpassingen of toevoegingen aan het basissysteem en zeer regelmatig maatwerkmodules. Andersom begint "maatwerk" niet met een lege terminal. Ook dan maken bedrijven gebruik van één of meer open source basissystemen of frameworks, commerciële "off the shelf" software of ten minste een verzameling bestaande modules die aan elkaar geschreven worden.

Daarnaast heeft elk stuk software configuratie en onderhoud nodig. Onderhoud zijn de wijzigingen die nodig zijn omdat kwetsbaarheden worden ontdekt of de systemen waarmee de software interacteert zich ontwikkelen. Tenslotte zal het geheel beschikbaar gemaakt moeten worden op een systeem. Wanneer dat via een netwerk gebeurt noemen we dat hosting (of de cloud). En uiteraard zal er een procedure moeten zijn voor backup en recovery. Dit kan ook gezien worden als een vorm van onderhoud, omdat continu getest moet worden of een systeem kan worden hersteld van backup. De eerste illustratie hierboven geeft de negen onderdelen weer die nodig zijn wil software functioneren in een organisatie. Een business model is een selectie van deze onderdelen of legt een accent.

Twee voorbeelden

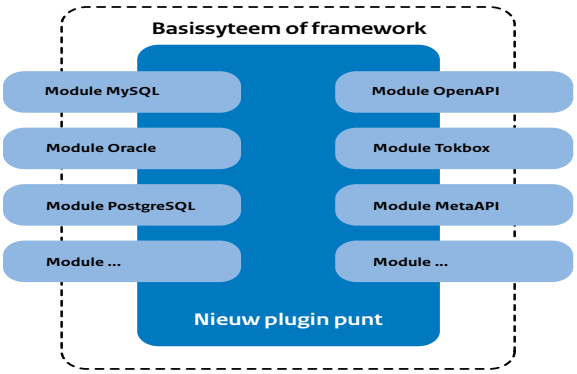
Om enig gevoel te krijgen voor deze negen onderdelen is het misschien goed om twee voorbeelden te geven. Het eerste voorbeeld hier zal ik tevens in hoofdstuk 7 gebruiken om de zaken concreet te maken.

Stel de opdrachtgever heeft de behoefte om gegevens in een database via het internet te ontsluiten als een Web API (of RESTful API). Dit is een actueel voorbeeld nu overheden steeds vaker gegevensuitwisseling real-time via het internet willen organiseren.

Heel simpel betekent dit dat je een vraag aan een database kunt stellen via een webadres. Stel je wilt de actuele bouwlocaties weten, dan vraag je dit op via een adres als bijvoorbeeld: website.nl/bouwlocaties/actueel

Deze manier van bevragen is uiteraard vooral interessant voor software. En met name software op andere locaties en/of bij andere organisaties. Meer precies is een API of *application programming interface* dan ook een verzameling definities op basis waarvan het ene stuk software kan communiceren met het andere stuk software[41]. Met een Web API gebeurt dit online en kan het ene stuk software informatie opvragen via een internetadres als hierboven geïllustreerd.

Dit vereist basissoftware, die enerzijds modules verbindt om toegang te verkrijgen tot de database en anderzijds modules die een definitie kunnen omzetten in een Web API. In de figuur rechtsboven is te zien dat de software als het ware twee sets "stopcontacten" heeft (punten van plug-in), waar verschillende modules als stekkers in kunnen. Eén plug-in punt voor de modules van input en één plug-in punt voor modules die deze gegevens weer ontsluiten.



Voorbeeld: software om gegevens via een online API te ontsluiten

Kunnen werkpakketten onafhankelijk van elkaar gerealiseerd worden?

De illustratie maakt tevens duidelijk dat deze modules *in principe* door verschillende leveranciers gemaakt kunnen worden. Dat zouden modules kunnen zijn die toegang geven tot een database die nu geen onderdeel zijn van dit basissysteem. Of het *reëel* is dat zo'n module los aanbesteed kan worden is afhankelijk van de vraag:

1. of de basissoftware daadwerkelijk een plugin punt bevat[42]
2. of het plugin punt in de basissoftware voldoende is gedocumenteerd en deze documentatie beschikbaar gesteld mag worden aan een andere leverancier
3. of de code van de basissoftware beschikbaar is, zodat als de module niet goed functioneert met de basissoftware de oorzaak kan worden gediagnosticeerd[43]
4. of de code van de basissoftware aanpasbaar is, zodat als een fout moet worden hersteld of een andere aanpassing nodig is, dit ook mogelijk is

Deze vier punten maken de leverancier van de module steeds onafhankelijker van de leverancier van de basissoftware. In principe zou reëel kunnen zijn om zo'n module door een andere leverancier te laten ontwikkelen als ten minste is voldaan aan de eerste twee punten. In de praktijk zal doorgaans ook het derde punt noodzakelijk blijken en zal het vierde punt de snelheid zeer ten goede komen.

> Maar in het algemeen: *hoe meer aan deze vier vragen wordt beantwoord, hoe reëler het is om de werkzaamheden over verschillende externe leveranciers te verdelen.*

Wanneer de code beschikbaar en aanpasbaar is, is het mogelijk om de basissoftware uit te breiden met een derde pluginpunt, die nog andere functionaliteit beschikbaar maakt. Bijvoorbeeld modules die een bepaalde wijze van toegang[44] implementeren. Dat vraagt om maatwerk aan het basissysteem zelf. Dit is het onderste oranje blok in de figuur hierboven.

[41] Een toegankelijke uitleg is hier te vinden: https://nl.wikipedia.org/wiki/Application_programming_interface.

[42] Je kunt zeggen dat deze zogenaamde "*plug in points*" API's op zichzelf, zijn, echter dat zou maar verwarring opleveren voor de lezer. Want zoals bij het proeflezen terecht werd opgemerkt: een Web API is niet noodzakelijk voor de uitbreidbaarheid. En zoals ook werd opgemerkt: er zou een gradatie van uitbreidbaarheid kunnen worden aangebracht aan de hand van deze termen. Echter, dit compliceert de zaak zonder dat dit bijdraagt aan het hoofdpunt in deze paragraaf: de negen onderdelen van elk softwaresysteem geven inzicht in hoe taken verdeeld kunnen worden over verschillende aanbieders en/of aan te besteden werkpakketten. Dat is wat het voorbeeld illustreert.

[43] Zoals door de proeflezers terecht werd opgemerkt is dit strikt genomen geen eis voor uitbreidbaarheid. Echter de vraag is hier of het *reëel* is dat een module los aanbesteed kan worden. Daarbij gaat het om de vraag naar afhankelijkheid. Ik breng hier gradaties aan van onafhankelijkheid. Bij dit punt: alleen als de aanbieder kan diagnosticeren of hijzelf of de leverancier van de basissoftware in gebreke blijft, kan hij de verantwoordelijkheid nemen voor de module, onafhankelijk van de aanbieder van de basissoftware. Het kunnen nemen van die verantwoordelijkheid is een voorwaarde voor het reëel zijn van het los aanbesteden van de module.

[44] Denk aan OpenAuth, Basic Authentication, Digest Authentication of vele andere.

Uiteraard heeft alle software onderhoud nodig. In dit voorbeeld blijkt dat bijvoorbeeld omdat van databases nieuwe versies uitkomen en dus de modules die toegang verlenen tot zo'n database onderhoud nodig zullen hebben. En uiteraard zal deze software in praktische zin toegankelijk gemaakt moeten worden. Dit is de hosting. Tenslotte zal het mogelijk moeten zijn om bij calamiteiten het systeem in korte tijd van kale machine terug te brengen in een werkzame toestand. Dit is backup & recovery.
De principes uit dit korte voorbeeld gelden voor elk systeem.

Meer over hosting

Software moet altijd ook beschikbaar gemaakt worden. Met hosting bedoelen we dat de infrastructuur beschikbaar is om de software beschikbaar te stellen. Tegenwoordig is dit doorgaans online via internet. Daarin zit een fysieke component van machines in een datacentrum en softwarematige component van besturingssystemen die onderhouden en gemonitord moeten worden. Het mag duidelijk zijn dat hoe meer gestandaardiseerd en meer universeel de hosting is opgezet, hoe makkelijker die overdraagbaar te maken is. Andersom kunnen er via de infrastructuur waarmee de software wordt ontsloten ook op allerlei manieren afhankelijkheden ontstaan. Zo zijn er cloud services die zogenaamde "microservices" aanbieden, zoals een database, die op een specifieke - niet generieke - manier benaderd moeten worden. Dit introduceert uiteraard exit kosten. In de praktijk schept dit een ongelijk speelveld wanneer er later opnieuw moet worden aanbesteed, al zouden exit-kosten in een aanbestedingscontext niet mee mogen spelen[45].

Hosting heeft de afgelopen jaren op allerlei manieren een nieuwe *"branding"* gekregen, onder namen als Private Cloud of "Infrastructure-as-a-service" (IaaS), "Platform-as-a-service" (PaaS), "Software-as-a-service" (SaaS). Deze laatste variant, is een businessmodel dat in feite alle negen elementen aan elkaar koppelt, inclusief de hosting als één product. We zien alle elementen die we hiervoor bespraken niet meer, we zien alleen het resultaat[46]. PaaS doet één stap terug en levert een deel van de configuratie en integratie, waar het gaat om ondersteunende software die het softwareproject nodig heeft om te functioneren, zoals het besturingssysteem, de database of de webserver. IaaS doet nog een stap terug en biedt gevirtualiseerde machines aan, al dan niet met een besturingssysteem of een specifieke "microservice".

Het mag duidelijk zijn dat hosting daarmee in elk geval sterk samenhangt met configuratie en integratie. Het is een heel andere vraag om het softwareproject werkend te realiseren op één (virtuele) machine, dan op een verzameling microservices. En zeker in dit laatste geval heeft het ook invloed op de architectuur van de software, die zal zich daarvoor moeten lenen.

In al deze "nieuwe" varianten van hosting zijn het niet de werkzaamheden die worden verricht om een bepaald resultaat te bereiken die als dienst worden aangeboden, maar de toegang tot de software is de dienst. Wat de werkzaamheden zijn om dat resultaat te bereiken wordt daarmee een black box. Of dat erg is, is vooral afhankelijk van de vraag of en in hoeverre die black box open moet wanneer de koper van leverancier moet wisselen. Want als de black box open moet bij migratie, dan zijn er exit kosten. Hosting, kortom, kan makkelijk leiden tot een technische verwevenheid, waaruit afhankelijkheden voortvloeien. Als de fysieke infrastructuur in privaat bezit is en de software is niet makkelijk verplaatsbaar, dan staat in praktische zin de software nog onder private controle. Als de kennis over de infrastructuur ontbreekt, waardoor niet duidelijk is wat er overgedragen moet worden, ook dan is dit een risico voor de continuïteit. Het gaat in dit geval niet (alleen) om afhankelijkheid die voortkomt uit juridische kwesties, maar om praktische afhankelijkheid.

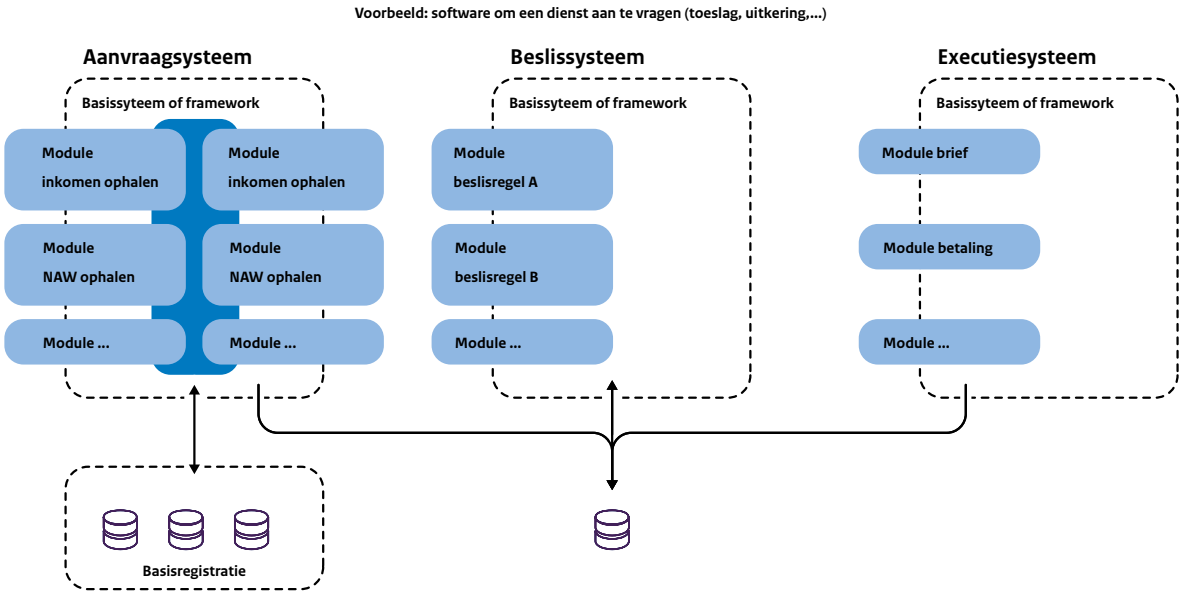
Er zijn ook hostingvarianten - die je tegenwoordig van het type "IaaS" zou noemen - waarmee het softwareproject in één of meer virtuele machine wordt geplaatst die verplaatst kunnen worden naar andere hostingproviders[47]. Zoals beschreven in het playbook open source aanbesteden is de clou vooral te zorgen dat precies dit een overdraagbare taak wordt. Door een specifieke configuratie op te laten leveren van virtuele machines en management software[48] daarvoor, wordt die omgeving onafhankelijk van de fysieke infrastructuur en daarmee overdraagbaar. Een heel concreet voorbeeld daarvan is Haven van Common Ground. Zij gaan nog een stap verder door hiervan een standaard te maken, die voor te schrijven en daarvoor ook tests te ontwikkelen. Er zijn daarom inmiddels allerlei leveranciers die "Haven compliant" een clouddiensten leveren.

[45] Paapst, 2013:93, Playbook Publieke Software Aanbesteden v2.0.
[46] Denk aan een abonnement op GoogleDocs, maar dit model is vrij universeel.
[47] Vandaag denken we vooral aan Docker, omdat dit een bredere bekendheid heeft gekregen, maar dit geldt al decennia voor allerlei soorten softwarecontainers die in (meerdere) virtualisatie-omgevingen kunnen draaien.
[48] Bij Haven is dit Kubernetis.

Een tweede voorbeeld: een aanvraagsysteem voor toeslagen of uitkeringen

Stel een overheid wil een systeem waarmee een burger online een aanvraag kan doen voor een dienst, zoals een uitkering of toeslag. Uiteraard is er iets nodig dat een formulier produceert voor de aanvraag, waarbij de burger tevens zijn gegevens uit de basisregistraties controleert.

Daarna oordeelt een set beslisregels over deze gegevensset en neemt het administratieve besluit. Tenslotte is er de fase van executie, waarbij de burger geïnformeerd wordt over het besluit en de betaling wordt uitgevoerd. De onderstaande figuur illustreert dit. In essentie implementeert elke uitvoeringsorganisatie dit proces.



Een manier om dit te modelleren is door dit te zien als drie basissystemen. Een eerste basissysteem ontsluit in essentie de basisregistraties en formulieren waarmee de burger de administratieve communicatie online kan voeren. Verschillende modules ontsluiten verschillende soorten formulieren.

Het tweede en derde basissysteem in dit voorbeeld, functioneren volgens dezelfde principes. De drie systemen zouden impliciet kunnen communiceren via een database-model, maar ook op andere manieren. Dit alles is afhankelijk van eerdere keuzes over de gewenste architectuur waaruit gewenste functionaliteit voortvloeit. Het punt is: elke vraag kan worden opgevat als een of meer iteraties van dezelfde verzameling van negen elementen. Afhankelijk van de rechten op het basissysteem[49] is het meer of minder makkelijk om aanvullende functionaliteit te laten ontwikkelen door andere leveranciers, aanpassingen te laten doen in het basissysteem en ook andere onderdelen af te splitsen in kleinere, losse offertes.

Vaak is één partij verantwoordelijk voor de integratie van al deze onderdelen tot een werkend systeem. Dit is doorgaans tevens de partij die verantwoordelijk is voor de hosting, omdat daar blijkt of het geheel functioneert. Maar zelfs dat is in principe te formuleren als een overdraagbare taak en een taak die meerdere organisaties gelijktijdig kunnen doen[50].

De essentie

1. Elke vraag kan worden opgevat als één of meer iteraties (of combinaties) van de verzameling van:
   - basissysteem
   - aanpassingen aan het basissysteem
   - (maatwerk)modules
   - onderhoud
   - configuratie of implementatie
   - hosting
   - backup & recovery.

[49] En afhankelijk van de vraag of er een stabiele en complete open standaard in dit basissysteem is geïmplementeerd.

[50] Net zoals in de meeste cloud-oplossingen meerdere dezelfde microservices draaien op verschillende systemen is het goed denkbaar om verschillende organisaties te laten werken aan een gezamenlijke procedure daarvoor, die ze gelijktijdig uitvoeren.

2. Een aanbesteding kan alleen geformuleerd worden in overdraagbare taken, indien er een visie is op de organisatie van de software en de onderlinge afhankelijkheden.
3. Als er een visie is op de organisatie van de software en duidelijk is wat de onderlinge afhankelijkheden zijn, dan kunnen in principe de voorwaarden geformuleerd worden waaronder elk van de negen elementen - zoals maatwerkmodules - onafhankelijk van elkaar aanbesteed kunnen worden. Althans in technische zin.
4. En eerder gaven we al aan dat het itereren over de publieke waarden antwoord geeft op de vraag of en waarom dat niet alleen praktisch, maar ook moreel of juridisch wenselijk en/of noodzakelijk is.

6 Strategische visie en marktonderzoek

Een strategische visie ontwikkelen, nodig voor de aanbesteding, is een iteratief proces, waarbij ten minste één keer wens en werkelijkheid met elkaar vergeleken worden. Software voor het primaire proces, zoals het nemen van administratieve besluiten, gegevensuitwisseling en data-analyse, is de infrastructuur van de samenwerking in de organisatie. Bij de ontwikkeling van een strategische visie moeten daarom enerzijds alle organisatorische belangen samenkomen. De reden daarvoor is dat keuzes in de architectuur overal consequenties kunnen hebben, waaronder functionele[51]. Deze kant van de visie valt buiten de scope van dit document, al hebben we in Hoofdstuk 5 Software wat kopen we dan? wel in essentie besproken hoe een softwareproject kan worden opgedeeld.

De andere belangrijke input voor de strategische visie komt voort uit de publieke waarden die zijn beschreven in Hoofdstuk 4 Waarom aanbesteden voor publieke waarden?. In de visie wordt beschreven *hoe* die waarden bereikt gaan worden. We nemen het voorbeeld uit het Hoofdstuk 5 weer in herinnering. In belangrijke mate valt in de strategische visie en de visie op de architectuur samen met de te realiseren waarden. Uit de visie in het voorbeeld van een basissysteem met enerzijds modules om te verbinden met verschillende databases en anderzijds modules om deze te ontsluiten, vloeit al een indeling voort in werkpakketten en een rudimentaire planning. Een basissysteem zal beschikbaar, gereed of gespecificeerd moeten zijn, willen de modules gemaakt kunnen worden. Een strategische visie die één of meer van de waardensets uit hoofdstuk 4 bevat en onderbouwt, leidt in combinatie hiermee tot een visie op het werkproces en daarmee ook de aanbesteding. Dat kan een werkproces zijn, waarbij verschillende aanbieders parallel kunnen werken aan de verschillende modules. Dat kan een werkproces zijn waarbij een integrator het basissysteem en de modules moet samenbrengen. Zo'n visie leidt tot eisen aan de rechten. In deze visie ligt oplevering onder een bestaande open source licentie voor de hand.

Maar uiteraard moet dit wel een reëel scenario zijn. Dit is een vraag die een marktonderzoek moet beantwoorden.

Het marktonderzoek

Gegeven deze input vraagt dit om een marktonderzoek[52]. Er zijn vele redenen waarom aanbieders van open source software en makers in den brede niet op dezelfde manier zullen en kunnen reageren op een marktconsultatie. Zo zijn de marges vaak anders, omdat zij niet of veel minder het rendement op intellectueel eigendom hebben, wanneer de software zelf onder een open source licentie wordt overgedragen. De omvang van de bedrijven is regelmatig kleiner. Bovendien is werken voor de overheid – en daarmee kennis opdoen van het werkproces bij de overheid – vaak de belangrijkste voorwaarde om succesvol een opdracht te kunnen verwerven. Het toepassen van een basissysteem of framework en toesnijden in een oplossing voor een (specifieke) overheidscontext vraagt bovendien een investering die vaak niet met rendement op intellectueel kapitaal kan worden terugverdiend. Dit zijn allemaal redenen waarom een marktonderzoek nodig is.
De vraag bij het marktonderzoek is enerzijds een vraag naar beschikbare software[53] en anderzijds een vraag naar de beschikbare bedrijven daar omheen.

- Zijn er bestaande systemen die – eventueel met aanpassingen en uitbreidingen – kunnen voldoen aan de functionele eisen? Of kan er een combinatie gemaakt worden van bestaande software?
- Zijn er bedrijven of ontwikkelaars beschikbaar rond die systemen? Zijn er bedrijven die bekend zijn met eventuele andere software die in combinatie nodig is? Of als dat allemaal niet het geval is: zijn er bedrijven die goed bekend zijn met de het soort software en tevens de taal waarin die geschreven is?

Welke bedrijven hebben daadwerkelijk bijgedragen aan de software die als basis zou kunnen dienen? Waar werken de ontwikkelaars die daadwerkelijk hebben bijgedragen?

[51] Het verhaal van het Finse kadaster kan daarbij als illustratie dienen: https://ibestuur.nl/podium/de-beste-bureaucratie-ter-wereld.

[52] Optioneel kan een marktconsultatie uitgevoerd worden. Soms is dat niet nodig, omdat er goed zicht is op het aanbod van de propriëtaire markt en de behoefte goed te verwoorden is. De wijze waarop een marktconsultatie gehouden kan worden en wat de overwegingen zijn zijn uitgewerkt op de website van PIANOo: https://www.pianoo.nl/nl/themas/innovatie/strategieen/stappenplan-marktconsultatie. Met dank aan Henk Jan Siersema die me hierop wees.

[53] Wanneer er reeds goed zicht is op het aanbod van de propriëtaire markt, dan gaat dit alleen om open source software. Zie ook de vorige voetnoot.

Dit zijn uiteraard vragen die een zekere expertise vragen. Bovendien vraagt zo'n onderzoek onafhankelijkheid. Wanneer hiervoor een externe organisatie ingehuurd wordt is het van belang goed te toetsen op deze onafhankelijkheid.

De uitkomst van het marktonderzoek

De uitkomst van het marktonderzoek zou antwoord moeten geven op in elk geval de onderstaande vragen. Elke van die vragen kan leiden tot een aanpassing of bijstelling van de strategische visie.

1. Welke architectuurbeslissingen zijn genomen in de verschillende oplossingen?
Sluiten die aan bij de eigen architectuurvisie? In welke mate zijn die te wijzigen? Het grote verschil tussen open source software en propriëtaire software is hier vaak dat propriëtaire systemen strengere eisen hebben voor het gebruik van bepaalde andere soft- en hardware. Als hierin maatwerk mogelijk is, dan komt dit maatwerk dan (vaak) terug bij updates en nieuwe releases. Ook de controle over de timing ligt vaak bij de bij aanbieder van propriëtaire systemen. Dit zijn risico's.
Welke (functionele) consequenties hebben deze beslissingen voor verschillende onderdelen van de organisatie?
Ook hier gelden de bovenstaande risico's.

2. Zijn er open source systemen of frameworks die een oplossing kunnen bieden?
a. Hoe breed en actief worden deze systemen onderhouden en ontwikkeld?
b. Zijn er bedrijven actief – wereldwijd – met ontwikkelaars die daadwerkelijk verstand hebben van de software? En zijn die bedrijven in principe bereid deel te nemen in een aanbestedingsprocedure?

3. Kan een systeem of framework worden samengesteld?
a. Is het reëel dat een markt ontstaat of geschapen kan worden voor dit nieuwe geheel?
b. Zijn er overheden met gelijksoortige behoeften?

4. De antwoorden op deze vragen kunnen aanleiding zijn om al voor de aanbesteding te reflecteren op de vraag waar compromissen gesloten kunnen of moeten worden met de belangen die in de oorspronkelijke visie centraal stonden. Dit kunnen tevens aandachtspunten zijn die later een rol kunnen spelen in de waardering van de gunningscriteria.

5. Een lijst op te stellen met risico-afwegingen, zoals:
Hoe zeker kan ik zijn dat onderhoud en ontwikkeling te koop is, tegen welke prijs op welke termijn?
Kan de beschikbaarheid van de software gediscontinueerd worden? Blijft de software beschikbaar in de vorm die ik voorzie?

6. Al voor de aanbesteding een keuze te maken voor open source software[54].

7. Eventueel om een pilotproject te starten om een proof of concept te kunnen testen.

[54] Er is dan geen aanbesteding nodig voor dit product. Dit heet ook wel het 'download-scenario'. De software kan de organisatie downloaden, daarnaast kan het nodig zijn om diensten aan te besteden.

7. Bringing it all together

Hieronder zullen we vier scenario's bespreken waarin we stap-voor-stap kijken naar elk van de negen elementen van een softwareproject en daarbij overwegen welke consequenties de sets van beginselen hebben voor het aanbestedingstraject.
In hoofdstuk 4 hebben we een aantal waarden benoemd, die afkomstig zijn uit de beleidsbrief "Open, tenzij". Dit zijn:

- continuïteit
- leveranciersonafhankelijkheid
- marktwerking
- samenwerking en ketensamenwerking
- innovatie en bedrijvigheid
- veiligheid en verantwoording

En hieraan hebben we ook transparantie toegevoegd als voorwaarde voor rechtsstatelijke waarden en met name het navolgen van het proces van besluitvorming en feitenverzameling.

Bij die waarden hebben we bronnen benoemd, die onderbouwen waarom het ook de taak of plicht is van elke overheid om die waarden na te streven[55].

Deze waarden zijn de belangrijkste input voor de strategische visie. De strategische visie geeft enerzijds weer hoe de software moet bijdragen aan het primaire proces van de organisatie. Anderzijds geeft de strategische visie ook weer hoe de zes waarden hierboven gerealiseerd zullen worden.

Deze stappen leiden tot *eisen* aan de rechten op software in een aanbesteding. Dat komt omdat deze rechten een voorwaarde zijn voor het realiseren van die waarden. Hieronder zullen we - aan de hand van die eisen - vier scenario's bespreken in volgorde van oplopende complexiteit. Uiteraard moet blijken uit het marktonderzoek dat dergelijke eisen een haalbare kaart zijn. Wanneer dit niet het geval is zal een scenario lager in deze lijst gevolgd moeten worden. De aanbesteding wordt dan ingewikkelder. Anderzijds kan de uitkomst van het marktonderzoek ook aanleiding zijn om in de strategische visie uit te werken hoe alsnog tot het gewenste scenario bereikt kan worden.

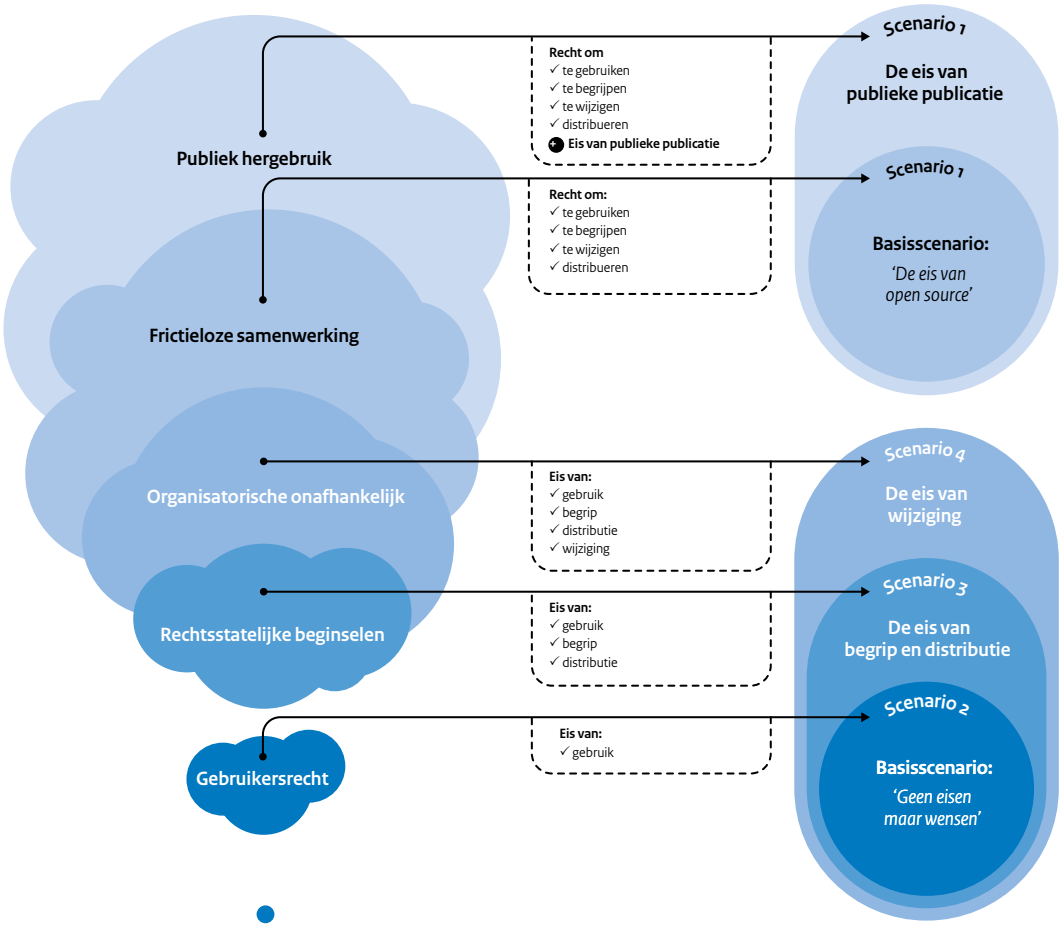
Hieronder zullen we de volgende vier scenario's bespreken:

1. de eis van een gestandaardiseerde open source licentie
2. geen eisen
3. de eis van inzage en distributie
4. de eis van gebruik, begrip, wijziging en distributie.

De twee extremen, het eerste en laatste scenario zullen we uitgebreid bespreken, omdat deze het meest verschillen. Dit zijn de basisscenario's. De andere twee scenario's zullen we korter bespreken in waar ze verschillen.

Deze scenario's bespreken we aan de hand van de negen onderdelen die noodzakelijk zijn wil elk softwaresysteem functioneren in de praktijk. Het eerste voorbeeld uit hoofdstuk 5 gebruiken we af en toe als illustratie.

[55] Daarmee kunnen overheden de argumenten verkrijgen die bepaalde eisen aan de rechten in een aanbesteding kunnen dragen. Als er voldoende argumenten zijn om een bepaald recht te vereisen, dan is zo'n eis draagkrachtig beargumenteerd. Dat is belangrijk omdat in een aanbestedingsprocedure aanbieders ook beargumenteerd alternatieven kunnen voorstellen, die niet zonder goede motivatie kunnen worden afgewezen.



Scenario's algemeen - hosting

In het voorbeeld hebben we al de hosting in het algemeen besproken. Daarin is aangegeven dat:

- configuratie en integratie van de software kan plaatsvinden op *virtuele* machines, die onafhankelijk zijn van de fysieke infrastructuur en daarmee van het eigendom van de fysieke infrastructuur;
- dat het van belang is dat dergelijke virtuele machines verplaatst kunnen worden naar andere hosting providers;
- dat de configuratie van het hosting- of cloudplatform zelf wordt opgeleverd als resultaat van de dienst, zodat het ook praktisch mogelijk is om de die virtuele omgeving te verplaatsen.

Bovenstaande maakt dat de configuratie en integratie van het eigen softwareproject daadwerkelijk onafhankelijk kan zijn van de hosting- of clouddiensten. Ze kunnen dan ondergebracht worden bij verschillende leveranciers. Omdat in het kader van dit document daarmee een natuurlijke grens is bereikt met de omgeving van het softwareproject, zal hieronder niet dieper worden ingegaan op het aanbesteden van de hosting- of clouddiensten zelf. Tegelijk gelden voor de hosting uiteraard dezelfde uitgangspunten die in dit document vaker naar voor komen, zoals overdraagbare taken en het zichtbaar maken van activiteiten.

Scenario's algemeen - backup & recovery

Backup en recovery zijn één geheel. Backup zonder de mogelijkheid tot recovery is betekenisloos. Toch komt het veel voor dat pas blijkt dat recovery niet regelmatig geoefend wordt, als de praktijk recovery noodzakelijk maakt.

Backup en recovery vereist dat het resultaat van alle stappen van integratie in hoog tempo opnieuw genomen kunnen worden. Dat betekent dat het basissysteem of framework beschikbaar moet zijn in de actuele versie, dat alle updates daarop toegepast kunnen worden, dat alle maatwerkaanpassingen beschikbaar zijn en daarop toegepast kunnen worden, de plaats van alle configuratiebestanden bekend is en beschikbaar in de laatste versie. En uiteraard moet alle data teruggezet kunnen worden.

Juist omdat er altijd onderhoud nodig is, heeft ook de backup & recovery *procedure* altijd onderhoud nodig. In de backup & recovery procedure ligt, als het goed is, buitengewoon veel kennis besloten over het softwareproject als geheel. Het is mede daarom wenselijk om dit door een andere organisatie te laten doen. Overdracht maakt expliciet welke handelingen uitgevoerd moeten worden, welke bestanden waar beschikbaar zijn en welke wijzigingen aangebracht zijn, althans als recovery geoefend en getest wordt.

Backup & recovery is het enige element dat in elk scenario - ook waarin geen bijzondere eisen worden gesteld aan de rechten - als een overdraagbare taak geformuleerd kan worden én gelijktijdig door meerdere organisatie uitgevoerd zou kunnen worden. Zij kunnen samen werken aan het opstellen, onderhouden en uitvoeren van de backup & recovery procedure. Zo wordt zekerheid niet via juridische weg verkregen, maar georganiseerd.

Scenario 1: vooraf kiezen voor open source software

In dit scenario leidt de strategische visie tot de conclusie dat een gestandaardiseerde open source licentie vereist is, eventueel in combinatie met publieke publicatie.

Het marktonderzoek

Uit het marktonderzoek is gebleken dat er bestaande software is die voldoet, er een bestaand ecosysteem van ontwikkelaars en bedrijven rond die software is of dat er ten minste ontwikkelaars en bedrijven zijn die diensten willen leveren rond deze software. Als dit niet het geval is, kan het vierde scenario een opstap zijn om die situatie te scheppen.

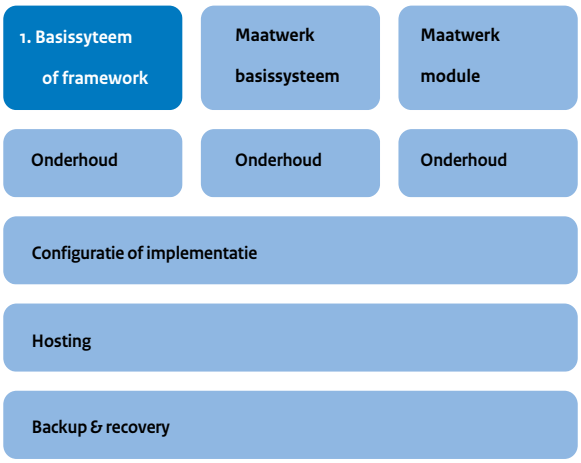
Het startpunt

- De visie beschrijft waarom het van belang is dat de software en de uitbreidingen daarop kunnen worden overgedragen tussen en aan leveranciers, toezichthouders, ketenpartners en andere overheden via publieke publicatie.
- Er is bestaande open source software, die voldoet of met aanpassingen kan voldoen aan de functionele eisen of het is juist de bedoeling software te ontwikkelen die publiek gepubliceerd wordt.

Kenmerkend voor dit scenario

- De negen elementen van het softwareproject kunnen als losse werkpakketten worden aanbesteed bij verschillende leveranciers.
- Er worden uitsluitend diensten aanbesteed.
- Het resultaat van die dienstverlening, wordt opgeleverd onder een open source licentie.

Het verwerven van de basissoftware I - de download-methode



Als het basisproduct reeds beschikbaar is als open source software, kan deze simpelweg worden verworven door deze te downloaden. Een aanbesteding is dan niet nodig. Dit heet ook wel de download-methode.

Software wordt soms gratis ter beschikking gesteld. Gratis betekent in de context van de aanbestedingswetgeving: zonder tegenprestatie. We associëren dat vooral met open source software, omdat die meestal publiek beschikbaar gemaakt[56] wordt. Als software publiek beschikbaar is én onder een open source licentie, dan is hij tevens gratis.

Aanbestedingswetgeving is niet van toepassing op het verwerven van gratis software. Verwerven van de software kan dan simpelweg door deze te downloaden.

Let op: In deze situatie is het wel van belang om te bedenken dat lang niet alle gratis software ook open source software is[57]. Gratis software kan ook propriëtaire software zijn, onder een licentie die (later) alsnog een tegenprestatie vereist bij specifieke vormen van gebruik. Verifieer daarom of er daadwerkelijk sprake is van een gestandaardiseerde open source licentie[58].

Omdat bij de verwerving de keuze *niet* aan de markt gelaten wordt, is er geen sprake van vergelijking van propriëtaire en open source software of licenties *in de aanbesteding*. In deze situatie is marktonderzoek belangrijk, omdat het niet vanzelfsprekend is dat bedrijven veel moeite zullen willen nemen om die functionaliteit te specificeren. Het basisproduct is immers onbetaald.

Het verwerven van de basissoftware II - (grotendeels) maatwerk

In bepaalde gevallen kan het juiste de bedoeling zijn om software (grotendeels) van de grond af te ontwikkelen en deze publiek ter beschikking te stellen. Denk bijvoorbeeld aan software die een bepaalde werkwijze mogelijk maakt die vanuit een publiek belang wenselijk is. Een voorbeeld daarvan is het vergelijken van data uit twee bronnen, zonder dat één van de twee bronorganisaties noodzakelijkerwijs de beschikking heeft over alle data[59]. Dit maakt een werkwijze mogelijk die breed in de publieke sector wenselijk is en overigens ook daarbuiten[60]. Publieke beschikbaarheid kan in zo'n geval bijdragen aan de verbetering, uitbreiding, adoptie en (de facto) standaardisatie.
In zo'n geval wordt een dienst aanbesteed, waarbij het resultaat wordt opgeleverd onder een (vooraf gekozen) open source licentie.

Let op: In zo'n geval is het belangrijk om goed na te denken over de specifieke wensen voor de rechtenstructuur. Is het juist wenselijk dat toevoegingen altijd publiek beschikbaar komen? Moet de software kunnen samenwerken met propriëtaire software?
Hoe verhouden de rechten zich met de architectuur[61]?

Scenario open source licentie vereist - alle andere onderdelen

Als de keuze voor het basissysteem of framework bekend is dan

1. worden daarnaast alleen diensten aanbesteed, geen producten
2. kunnen in principe alle andere acht diensten onafhankelijk in verschillende werkpakketten worden aanbesteed.

Het is daarbij wel van belang om een visie te hebben op de samenwerking (zie ook: na de aanbesteding). In bestaande open source projecten worden ontwikkelaars vaak gesocialiseerd in een bepaalde werkwijze. Dat wil zeggen dat ze door de tijd leren wat de regels zijn voor de samenwerking. Die regels blijven vaak impliciet. Het kan lonen - als er een bestaand ecosysteem van ontwikkelaars en bedrijven is - om die regels expliciet te maken of laten maken. Wanneer er geen bestaand ecosysteem is, om te bedenken hoe die regels zouden kunnen luiden. Het gaat dan om zaken zoals:

1. **De plaats van bestanden**
   Waar staan de tests, waar staat de documentatie, de updatescripts, etc.
2. **De normen voor documentatie**
   Hoe ziet de functionele documentatie eruit, waar staat de roadmap, hoe worden subroutines gedocumenteerd, etc.

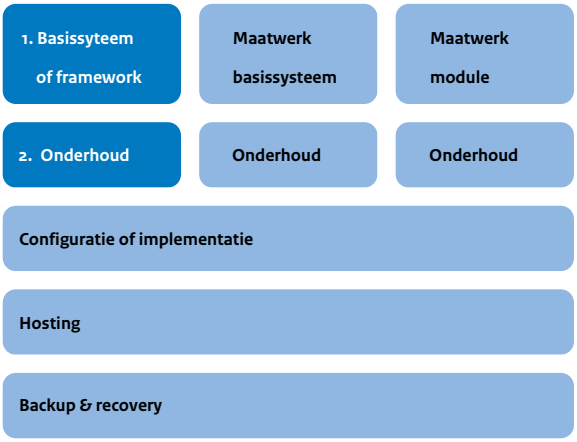
[56] Op een eigen website of via een platform als sourceforge.net, github of gitlab.
[57] Andersom is ook er geen belemmering om voor open source software een tegenprestatie in geld te vragen, ook niet onder de GPL. In dat laatste geval is het uiteraard wel zo dat de verkrijger daarna de software mag herdistribueren zonder verdere kosten. Zie: https://www.gnu.org/licenses/gpl-faq.html#DoesTheGPLRequireAvailabilityToPublic.
[58] https://opensource.org/licenses/category, https://www.gnu.org/licenses/license-list.html en over de verwarring die vaak optreedt: Playbook Publieke Software Aanbesteden v2.0.
[59] Meer concreet: multiparty secure computation.
[60] De voordelen van het delen van gegevens botsen dan niet meer met (een deel van) de nadelen daarvan.
[61] Dit laatste kan voor bijzondere wensen heel relevant zijn. Zie bijvoorbeeld: https://www.iusmentis.com/computerprogrammas/opensourcesoftware/mixedsource/#Juridischontwerpvansoftware.

3. **De normen voor tests**
   Welke testscenario's zijn er, hoe wordt elke subroutine getest, hoe wordt het upgraden van data getest, etc.
4. **De normen voor versies en upgrades**
   Welke versies worden onderscheiden, hoe werken de upgradescripts, wie brengt de basisdistributie uit bij elke versiewijziging, etc.

Het zijn vaak logische en praktische elementaire regels. Hoe duidelijker dergelijke regels zijn, hoe makkelijker en efficiënter mensen en bedrijven samenwerken.
Hoe duidelijker zo'n werkwijze is, hoe makkelijker het ook is om code over te dragen, een andere partij code te laten installeren of integreren. Het is dan zelfs mogelijk om veel van deze normen geautomatiseerd te toetsen.

Wie deze normen vooraf documenteert, kan ze gebruiken als onderdeel van het aanbestedingsproces. En belangrijker nog: later, als meer routinematig kleinere opdrachten onder meerdere partijen worden weggezet. Partijen conformeren zich dan aan een werkwijze als onderdeel van de aanbesteding.

Scenario open source licentie vereist - onderhoud



Alle software is permanent in ontwikkeling en vereist permanent onderhoud. Software staat nooit op zichzelf en moet aansluiten op de ontwikkeling van software in zijn omgeving, zoals databases of besturingssystemen. Daarnaast zijn er altijd bugs die gevonden worden en veiligheidsproblemen die verholpen moeten worden. Dit is onderhoud.

- Meerdere bedrijven kunnen maximaal onafhankelijk van elkaar onderhoudsdiensten leveren, omdat de software beschikbaar is onder een gestandaardiseerde open source licentie
- Verschillende vormen van onderhoud, kunnen door verschillende bedrijven uitgevoerd worden
- Dezelfde vormen van onderhoud kunnen door meerdere bedrijven uitgevoerd worden, gelijktijdig of roulerend
- Het is altijd mogelijk om te toetsen of en welke kwetsbaarheden in de code zitten, want de code is beschikbaar
- Het is altijd mogelijk om te toetsen of bekende oplossingen voor (veiligheids)problemen (tijdig) worden toegepast

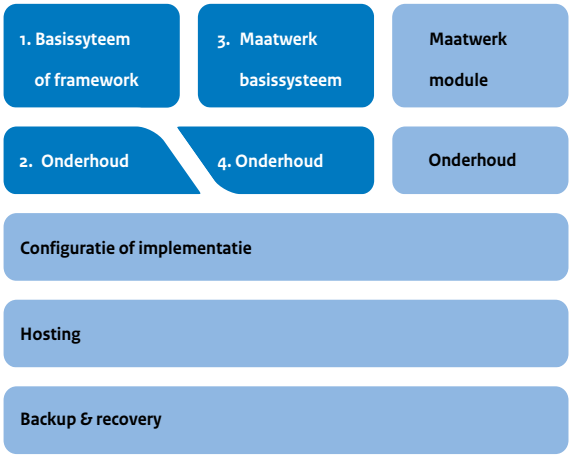
Elk van de bovenstaande mogelijkheden kunnen onderdeel worden van de aanbestedingsstrategie van dit werkpakket.

Omdat alle leveranciers toegang hebben tot de broncode en deze ook mogen wijzigen en verspreiden is het in dit scenario mogelijk om zowel meerdere bedrijven onderhoudsdiensten te laten leveren, als verschillende vormen van onderhoudsdiensten te laten uitvoeren door verschillende leveranciers. Bij dat laatste kun je bijvoorbeeld denken aan onderhoudsdiensten in de vorm van het verhelpen van bugs en kwetsbaarheden enerzijds en het monitoren van bekende kwetsbaarheden en het toetsen of en op welke termijn deze updates daadwerkelijk zijn doorgevoerd. Dit zijn rollen die kunnen rouleren, zodat bedrijven scherp gehouden worden, zonder prikkel om elkaar in een kwaad daglicht te stellen. Op deze manier kan niet alleen een grote mate van leveranciersonafhankelijkheid worden verworven, maar gaat ook de kwaliteit van het proces omhoog op een transparante en verifieerbare manier.

Het probleem, dat niet getoetst kan worden welke bestaande en bekende kwetsbaarheden er in de code zitten en of bekende oplossingen (tijdig) worden toegepast zijn, speelt niet. De code zoals die gebruikt wordt is immers beschikbaar.

Naarmate er minder een bestaand ecosysteem is van ontwikkelaars en bedrijven is het belangrijker om het onderhoud te verspreiden over verschillende bedrijven. Op die manier kan de markt geschikt(er) gemaakt worden voor de doelstellingen die nagestreefd worden. Juist onderhoud leent zich hiervoor, omdat dit meerdere kleinere opdrachten kunnen zijn en deze bedrijven samen moeten werken met de core-contributors, vaak de oorspronkelijke makers van de software.

Scenario open source licentie vereist - maatwerk basissysteem en onderhoud



Ook maatwerk aan het basissysteem kan in dit scenario onafhankelijk worden aanbesteed van andere onderdelen. Het is ook zeer wel denkbaar om het toepassen van dat maatwerk op het basissysteem aan een andere partij te laten, dan het onderhoud en het onderhoud aan het maatwerk op het basissysteem. Zo kan de documentatie en functionaliteit getest getest worden door een andere organisatie, wat de overdraagbaarheid vergroot en bewijst.

De mogelijkheid om maatwerk deels te abstraheren en te integreren in het basissysteem neemt toe wanneer meerdere organisaties bekend zijn met zowel de basiscode als de maatwerkcode. Dit kan de onderhoudslast van de maatwerkcode reduceren.

Maatwerk en de basisdistributie

Naarmate het bestaande ecosysteem kleiner is, is het beter mogelijk de basisdistributie te controleren via het opdrachtgeverschap of een afsplitsing (fork) daarvan zelf te controleren. Er zal wellicht intuïtief een neiging bestaan om dit te doen, omdat de focus ligt op de eigen vraag, de eigen functionaliteit en de eigen opgave. Toch is het de vraag of dit wenselijk is. Net zoals overheden afhankelijk kunnen worden van leveranciers, kunnen ze ook afhankelijk worden van hun eigen maatwerk of de kennis bij enkelen in de organisatie van dat maatwerk. Veel verstandiger is om het maatwerk te blijven ontwikkelen in dezelfde cyclus als de basisdistributie. Nog beter is om maatwerk zoveel mogelijk te abstraheren zodat die onderdeel kan worden van de basisdistributie en een configuratie wordt of te modulariseren. Zie ook "na de aanbesteding".

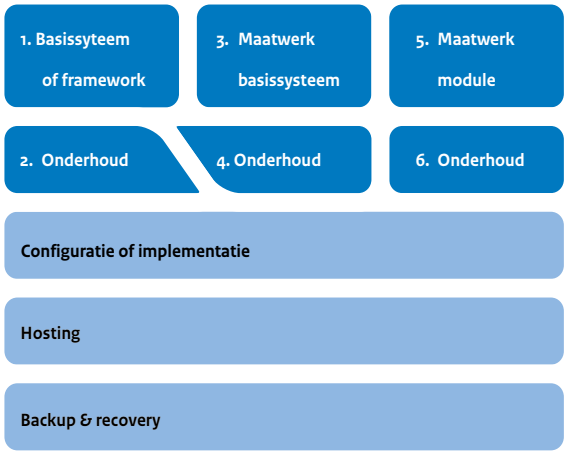
Overdragen via publicatie

Om optimaal te profiteren van frictieloze overdracht kan ook de maatwerkcode onder dezelfde open source licentie worden opgeleverd. Al is die code minder interessant voor derden, verschillende leveranciers kunnen dan frictieloos de code tot zich nemen en gebruiken. Ook maakt dit het waarschijnlijker dat de maatwerkcode z'n weg vindt - meer of minder geabstraheerd - naar de basisdistributie.

Maatwerk en inkoopvoorwaarden

Maatwerk opleveren onder een open source licentie vraagt in de praktijk een uitzondering op de inkoopvoorwaarden. In de praktijk is het buitengewoon handig om een geheel als resultaat op te laten leveren en dit botst met de meest gebruikte open source licenties[62].

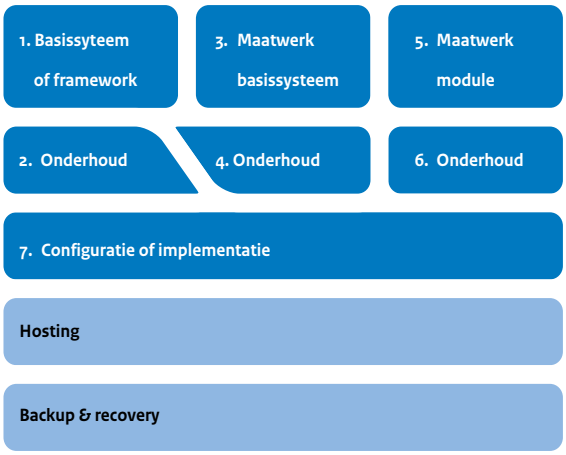
Scenario open source licentie vereist - maatwerk-module en onderhoud



Maatwerkmodules kunnen in dit scenario onafhankelijk worden aanbesteed van andere onderdelen. Het is verstandig om ook een maatwerkmodule zoveel mogelijk te abstraheren, zodat die onderdeel kan worden van de basisdistributie, ook wanneer het onderhoud daarvan volledig bij de eigen organisatie terecht komt.
Het voordeel hiervan is dat nieuwe versies van de basisdistributie dan getest worden met de zelf ontwikkelde module. Fouten komen hierdoor eerder aan het licht.
Het is zeer wel denkbaar om de realisatie van de module, de installatie en configuratie en het onderhoud in verschillende werkpakketten en bij verschillende leveranciers onder te brengen. Ook hier geldt dat dit een toets is voor de documentatie en de overdraagbaarheid. Bovendien verspreidt de kennis van de code zich daarmee over meerdere aanbieders.

[62] Zowel ARBIT (artikel 8 in versie 2018) als ARVODI (artikel 24 in versie 2018) vereisen in principe volledig eigendom van de resultaten op het moment dat die ontstaan. Dat botst met de mogelijkheid om een aangepast *geheel* op te leveren wanneer de licentie vereist dat die onder dezelfde voorwaarden worden geleverd.

Scenario open source licentie vereist - configuratie en integratie



Met integratie wordt bedoeld dat alle onderdelen bij elkaar worden gebracht, geïnstalleerd en geconfigureerd, zodat ze werken als één systeem[63]. Een basissysteem is - als het goed is - zoveel mogelijk geabstraheerd. Dat wil zeggen dat dit systeem allerlei generieke elementen bevat, die geselecteerd moeten worden of ingesteld. Denk daarbij aan de vraag welke modules actief zijn, het verbinden met de database of het instellen van formulieren.

In ons voorbeeld van het ontsluiten van een database via internet zal de module voor de specifieke database gespecificeerd moeten worden. Ook staat software vrijwel nooit op zichzelf, maar moet samenwerken met andere software - zoals een webserver. Regelmatig is wat we maatwerksoftware noemen in werkelijkheid een uitgebreide vorm van configuratie en integratie.

Ook hier geldt dat deze dienst zoveel mogelijk als een overdraagbare taak geformuleerd moet worden. Dat betekent in elk geval het opleveren van de configuratie bestanden en het documenteren van het werkproces als een overdraagbare routine. Vaak gaat dit gepaard met zogenaamde *"tooling"* waarmee wordt bedoeld dat het werkproces wordt ondersteund met scripts die bepaalde onderdelen geautomatiseerd uitvoeren. Ook dit zijn op te leveren resultaten van de dienst. Hoe beter dit gebeurt, hoe reëler het wordt om deze dienst te laten rouleren over meerdere leveranciers[64]. Want ook dit is geen eenmalige taak, maar een continu proces. Dat komt omdat het systeem, het maatwerk daaraan en de maatwerkmodules onderhouden worden.

Scenario 2: geen eisen, maar wensen

In dit scenario leidt de strategische visie tot de conclusie dat er geen enkele andere eis gesteld moet worden aan de rechten op de software, dan het gebruiksrecht.

Het marktonderzoek

Op deze conclusie heeft een marktonderzoek gevolgd, waaruit is gebleken dat er bestaande software is die voldoet, of kan voldoen met aanpassingen, en dat er bestaande leveranciers zijn die diensten willen leveren rond dergelijke software. Elke bestaande oplossing die beschikbaar is als open source software, is functioneel minder compleet. De wens om alle aangeboden oplossingen te vergelijken op beste prijs-kwaliteit van het aanbod wordt daarom verkozen boven de download-methode.

Het startpunt

Dit scenario kenmerkt zich door deze factoren:

- Er is bestaande software die voldoet of met aanpassingen kan voldoen aan de functionele eisen.
- Er zijn geen bijzondere eisen aan de rechten.

De visie beschrijft *niet* de noodzaak om software en uitbreidingen daarop over te dragen tussen leveranciers, maar overweegt de mogelijkheid van overdraagbare taken. (hosting en backup & recovery zijn reeds generiek beschreven)

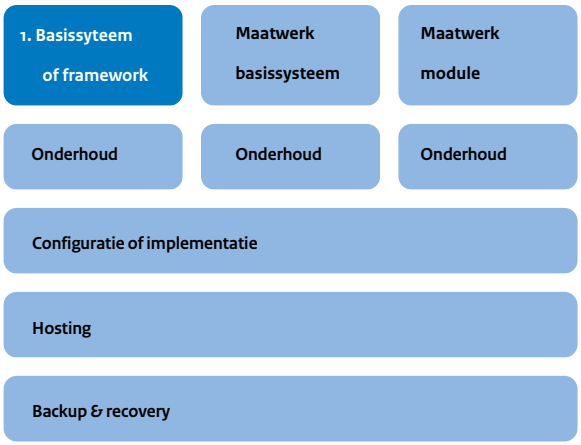
Kenmerkend voor dit scenario

- De keuze tussen propriëtaire software en open source software wordt aan de markt gelaten.
- De voorkeur voor open source software kan tot uitdrukking via een hogere waardering voor ruimere gebruiksrechten.
- Diensten en producten worden deels gebundeld ingekocht, omdat vooraf niet zeker is dat ze onafhankelijk van elkaar uitgevoerd kunnen worden.

[63] Daarbij kan ook de migratie van data horen, maar dit is buiten scope.

[64] Meer in het algemeen zouden partijen kunnen rouleren in pools voor bijvoorbeeld nieuwe code, onderhoud en configuratie & integratie. Dit zorgt ervoor dat documentatie en geautomatiseerde tests altijd twee keer worden overgedragen en getest. De organisatie(s) die verantwoordelijk is (zijn) voor de configuratie en integratie heeft daarbij een relatieve machtspositie. Het rouleren van bedrijven in verschillende rollen zorgt dat het minder aantrekkelijk is om misbruik te maken van die positie.

Het verwerven van de basissoftware - keuze aan de markt



In deze scenario's onderscheiden we het verwerven van een licentie op een bestaand *product* van de andere elementen in een project. Bij alle andere acht elementen verwerven we geen product, maar *diensten*[65]. We laten die diensten kortom geen *impliciet* onderdeel van het product. Aan deze licentie stellen we alleen als *eis* het gebruiksrecht, want zonder gebruiksrechten kunnen we niets met dit product.

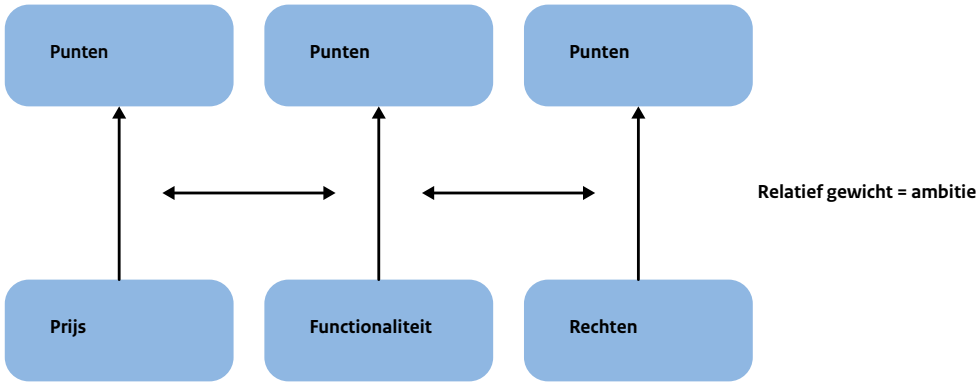
Uitgaande van een situatie waarin een potentieel product (ook) onder een open source licentie beschikbaar is, kunnen de rechten als één van de kwaliteitscriteria worden meegewogen, zoals geïllustreerd in de onderstaande figuur.

Beoordeling mede aan de hand van beschrijving van de leverancier

Wanneer de keuze aan de markt gelaten wordt, kan de aanbestedende organisatie alternatieven mede beoordelen aan de hand van de beschrijving en beoordeling van de leverancier. Echter wanneer het product als apart werkpakket zou worden aanbesteed, en niet gebundeld met aanvullende diensten, dan zal er voor een aanbieder van open source software weinig tot geen incentive zijn om die tijdsinvestering in een aanbesteding te doen. Tenzij bepaalde onderdelen gelijktijdig als taak - en dus als dienst - worden geformuleerd is het doorlopen van een aanbestedingsprocedure voor aanbieders van open source software een last zonder beloning.

Nu is dat in dit scenario een theoretische situatie. Dat komt omdat vooraf niet zeker is dat voldoende rechten worden verkregen om andere diensten onafhankelijk van de leverancier van het product aan te besteden. Eerst het product aanbesteden en daarna de diensten is om vele redenen ook geen optie. Zo kan een propriëtair product niet als vereiste worden opgenomen in een aanbesteding.
De maker heeft een enorm voordeel en elke andere aanbieder zou afhankelijk zijn van de eigenaar van het product. Het product zal daarom - uit noodzaak - gebundeld met de meeste dienstverlening moeten worden aanbesteed, zoals uit de overwegingen bij de verschillende onderdelen hieronder zal blijken. Het voordeel hiervan is wel, dat nu ook leveranciers van open source software een incentive hebben om het product te beschrijven aan de hand van de wensen in de aanbesteding. Zij kunnen hun marge maken op de diensten die gebundeld worden aanbesteed.

Gunningscriteria



[65] In de situatie dat we een licentie op een module verwerven, al dan niet met maatwerk aan die module, dan is dit een nieuwe iteratie door deze negen elementen.

Gunning mede op basis van ruim gebruiksrecht

Als de keuze aan de markt wordt gelaten, worden rechten niet vereist, maar gewenst. Aanbieders worden niet alleen gewaardeerd op prijs en functionaliteit, maar ook op de rechten die ze aanbieden. Er is niets tegen om bij de weging meer punten toe te kennen aan een inschrijving met een ruim gebruiksrecht. In de afbeelding hierboven is het principe te zien waarbij invulling wordt gegeven aan de algemene strategie van de zogenaamde "beste prijs-kwaliteitverhouding" (BPKV).

Belangrijk: Hoewel Pianoo op haar website rechten niet expliciet benoemt als subgunningscriterium[66], is dit niet anders dan bijvoorbeeld milieukenmerken, sociale factoren of risicomanagement. Net zoals van duurzaamheid een gunningscriterium gemaakt kan worden, van kwaliteit, een plan van aanpak of de waardering in het verleden, geldt dit ook voor meer of minder ruime gebruikersrechten. Afhankelijk van de specifieke ambitie of het algemene ambitieniveau kan hieraan een groter relatief gewicht gegeven worden.

Hieronder is een voorbeeld te zien van hoe zo'n gunningscriterium eruit kan zien. In ons voorbeeld van een basissysteem voor het ontsluiten van een database via het internet zou een functionele de wens kunnen zijn dat de software zoveel mogelijk database-onafhankelijk is en het zowel closed source als open source databases kan ondersteunen. De reden hiervoor kan gelegen zijn in flexibiliteit en leveranciersonafhankelijkheid.

Voorbeeld: functionele wens ten behoeve van overdraagbaarheid (bron: NOiV 2009)

Wens	Wijze van punten-toekenning	Punten
De software is zoveel mogelijk database-onafhankelijk en koppelbaar met databases van meerdere leveranciers, waarbij het de voorkeur geniet dat de software koppelbaar is met zowel closed source als open source databases. De reden hiervoor is gelegen in flexibiliteit en leveranciersonafhankelijkheid. Deze wens wordt beoordeeld op de mate waarin c.q. de wijze waarop invulling wordt gegeven aan het gestelde.	Software koppelbaar met één database	0
	Software koppelbaar met ofwel meerdere closed source databases, ófwel meerdere open source databases	4,5
	Software koppelbaar met meerdere databases, waarvan er in elk geval één opensource en één closed source is	9

Afspraken, die als recht (product) of als taak (dienst) kunnen worden ingevuld

Er zijn ook nog andere afspraken die als recht *kunnen* worden ingevuld. Denk daarbij aan garanties, aansprakelijkheid en vrijwaring. Als dat gebeurt, is het zaak om te zorgen voor een gelijk speelveld. Zo is het gemakkelijker om een garantie af te geven dat de software geen rechten van derden schendt als de broncode niet (publiek) beschikbaar is, omdat schendingen veel lastiger kunnen blijken. Juist omdat het hier gaat om het verlenen van zekerheden kan een hogere waardering gegeven worden wanneer dit objectief en transparant - en dus zeker - nagegaan kan worden.

Het invullen van gewenste zekerheden met rechten[67] leidt makkelijk tot schijnzekerheid, waarbij in feite de vraag is welke leverancier het meeste lef heeft om een specifieke garantie voor de afnemer als (financieel) risico op zich te nemen als leverancier. Beter is om de vraag te stellen wat het werkproces is dat een bepaalde ongewenste situatie helpt voorkomen en - als het toch voorkomt - wat de problemen die dan ontstaan helpt op te lossen. Nog beter is het wanneer die werkprocessen ook door organisaties anders dan de leverancier uitgevoerd kunnen en mogen worden (overdraagbare taken). Hieronder twee voorbeelden.

[66] https://www.pianoo.nl/nl/metrokaart/beste-prijs-kwaliteitverhouding.

[67] Of dit nu een garantie, vrijwaring of aansprakelijkheid is, de koper verkrijgt een recht op iets onder bepaalde omstandigheden. Daarom spreek ik over rechten. Tegenover dat recht van de koper staat een plicht van de leverancier. Meestal is dat een financiële vergoeding. Een dergelijke overeenkomst leent zich voor juridische conflicten, die in zo'n situatie eerder afleiden van het werkelijke probleem dat is ontstaan, dan dit probleem oplossen.

Voorbeeld 1 (geen) escrowregeling

Om faillissementsrisico af te dekken worden wel een zogenaamde escrowregeling afgesproken[68]. Een "gespecialiseerd escrow agent" neemt de code dan in bewaring. In veel gevallen gaat het dan om het deponeren van code in een (fysieke) kluis[69] en het maken van juridische afspraken over afgifte- en gebruiksrecht tussen drie partijen[70]. In de praktijk is dit uiteraard slechts een beperkte voorwaarde voor het afdekken van het werkelijke risico. Wat heb je aan code zonder kennis en de beschikbaarheid van organisaties die de dienstverlening potentieel kunnen overnemen?

> In het eerste scenario komt dit onderwerp niet naar voren, omdat het niet relevant is. Wanneer meerdere partijen niet alleen reeds beschikking hebben over de code, maar daarnaast ook *werken* met de code, is een faillissementsrisico reeds met grotere zekerheid afgedekt, dan met een escrowregeling bereikt kan worden.

Voorbeeld 2 het schenden van rechten van derden

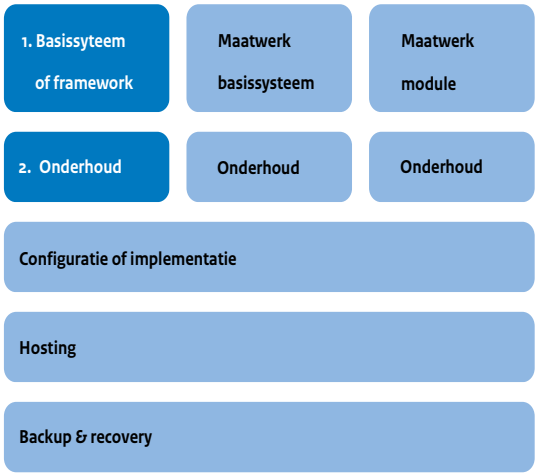
Een tweede voorbeeld is het schenden van rechten van derden. Als dit in een aanbesteding gewaardeerd wordt, kan de vraag zijn of de code door één of meerdere organisaties actief is bekeken en of dit extern te verifiëren is. Of denk aan de mogelijkheid om te zien of de betwiste code door één of meerdere auteurs is geschreven, op welk moment de code is toegevoegd en of nagegaan kan worden welke code deze auteur nog meer toegevoegd heeft. Gebruik van een versiebeheersysteem kan hoger worden gewaardeerd, nog hoger wanneer meerdere organisaties hierin samenwerken, nog hoger wanneer dit publiek beschikbaar is en de code vergeleken kan worden. Ook hier gaat het om de vraag of dit objectief en transparant te verifiëren is.

Als zou blijken dat er inbreuk is gemaakt op de rechten van derden, dan is vooral de vraag of het binnen de mogelijkheden ligt van de leverancier om dit probleem op te lossen. Heeft hij zelf toegang tot de broncode en het recht, de mogelijkheden en de kennis om die te wijzigen, zodat hij dit probleem kan oplossen? Zijn er andere organisaties die bij falen dit probleem kunnen oplossen? Dit zijn vragen die zich richten op het werkelijk belang van de koper.
In bovenstaande voorbeelden praten we uitsluitend over het product zelf. Leveranciers van propriëtaire software leveren vaak diensten *als onderdeel* van hun product, dat komt *mede omdat* ze zich laten betalen voor de licentie[71].

Bij leveranciers van open source software is product al vrij beschikbaar en zijn dergelijke garanties vaak al georganiseerd zijn in het werkproces van open source software. Voor een gelijk speelveld is het dus enerzijds belangrijk om de diensten die we vaak als onderdeel van het product zien expliciet te maken en te waarderen. Anderzijds is het belangrijk voor een gelijk speelveld dat de dienstverlening in de andere acht elementen van het project gelijktijdig wordt ingekocht.

Die andere elementen bespreken we hieronder. Dan zal blijken dat het gelijktijdig inkopen van veel van die onderdelen in één aanbesteding om heel andere redenen ook noodzaak is bij aanbieders van propriëtaire software. Zeker in dit scenario.

Scenario *geen eisen* - onderhoud



> Alle software is permanent in ontwikkeling en vereist permanent onderhoud. Software staat nooit op zichzelf en moet aansluiten op de ontwikkeling van software in zijn omgeving, zoals databases of besturingssystemen. Daarnaast zijn er altijd bugs die gevonden worden en veiligheidsproblemen die verholpen moeten worden. Dit is onderhoud.

We verwachten allerlei vormen van dienstverlening naast het "off the shelf" product of basissysteem zelf. Op de eerste plaats allerlei vormen van onderhoud. Het gaat dan bijvoorbeeld om fouten (bugs), die soms kwetsbaarheden kunnen zijn. Maar ook wijzigingen die nodig zijn, omdat de

[68] Zie bijvoorbeeld: https://www.pianoo.nl/nl/inkoopproces/fase-1-voorbereiden-inkoopopdracht/opstellen-selectiecriteria/faillissementsrisico#Beperkenvandemogelijkegevolgenvanfaillissement.
[69] Zie bijvoorbeeld: https://www.escrowalliance.com/nl/escrow-regelingen/broncode-escrow-regeling/.
[70] Zie bijvoorbeeld: https://nl.wikipedia.org/wiki/Escrow-overeenkomst.
[71] Denk aan artikel 7:17 BW (Conformiteit).

software gebruik maakt van andere software - denk aan databases uit ons voorbeeld - die zich ontwikkelt en daarom ook weer eisen kan stellen aan de software die er gebruik van maakt. Vaak veronderstellen we dit soort activiteiten als impliciet onderdeel van het product. Het is om verschillende redenen belangrijk om dit soort activiteiten expliciet te maken.

Lang niet alle softwareleveranciers publiceren gevonden fouten en kwetsbaarheden. Niet altijd zijn er afspraken te maken over wanneer nieuwe versies (moeten) worden uitgerold. Bij open source software worden fouten en kwetsbaarheden en hun oplossingen (vrijwel) altijd gepubliceerd[72]. Dergelijke verschillen hebben consequenties, die gewaardeerd kunnen worden in gunningscriteria.

Commerciële software bevat zo goed als altijd voor een groot deel open source software[73]. Dit is echter niet altijd zichtbaar bij propriëtaire software. Ook niet als dit open source software onder een toegeeflijke licentie was, die met aanpassingen opnieuw is uitgegeven onder een propriëtaire licentie[74]. Het is daarmee ook niet zichtbaar of oplossingen voor gevonden kwetsbaarheden in de gebruikte code in het aangepaste product zijn verwerkt en indien wel, na welke termijn. Bij open source software is dit *by default* (extern) toetsbaar. Hoewel publieke beschikbaarheid van de code *op zichzelf* voldoende noch noodzakelijke voorwaarde om dit te toetsen. Wel is het veel eenvoudiger om een procedure, zoals een audit, op te stellen, die als een overdraagbare taak kan worden ingekocht. Het is precies dit dat gewaardeerd kan worden in een aanbesteding aan de hand van gunningscriteria. Denk daarbij aan vragen als:

- in welke mate is toetsbaar en transparant welke kwetsbaarheden bekend zijn.
- in welke mate *kan* toetsbaar en transparant gemaakt worden hoe met het registreren van kwetsbaarheden wordt omgegaan.
- op welke wijzen kunnen updates beschikbaar gemaakt worden?
- welke mate van controle heeft de organisatie over de installatie van updates?

Gelijk speelveld
Zorg voor een gelijk speelveld tussen propriëtaire en open source software, door alle mogelijkheden objectief en transparant te waarderen

De wet zegt dat het verhelpen van fouten altijd mogelijk is, ook zonder toestemming van de maker. Ook is het mogelijk om daarvoor derden in te huren[75]. Maar dit is theorie. Een werkwijze zonder broncode en zonder medewerking van de maker is verre van ideaal. Het is een noodgreep, omdat dit bestudering van de software vergt en/of decompilatie. Onderhoud aan het basissysteem zal in dit scenario dan ook in één aanbesteding moeten worden verworven met het product, omdat niet tevoren duidelijk is dat dit onafhankelijk kan worden uitgevoerd.

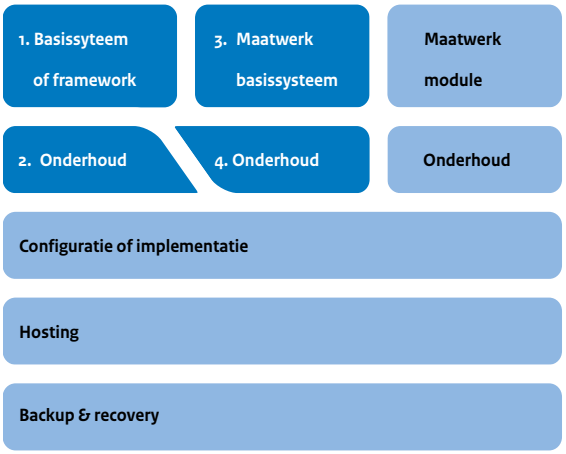
Risico's
Risico's die we als onvermijdelijk zien, benoemen we meestal niet als risico, maar als ze in vergelijking niet onvermijdelijk zijn, dan moeten ze gewaardeerd worden.

Als het basissysteem door slechts één partij onderhouden kan worden, dan is er sprake van een grote afhankelijkheid wanneer het gaat om het verhelpen van een kwetsbaarheid en de tijdspanne waarbinnen dit gebeurt. Ook voor informatie over de risico's die de organisatie precies loopt en hoe die beperkt kunnen worden is de verwerver dan in belangrijke mate afhankelijk van de leverancier. Hoe - en bij wie - de verwerver kan rekenen op (informatie over) onderhoud en welke alternatieven daarvoor zijn kan een rol spelen in de gunningscriteria.

Tenslotte is ook de dwingendheid van de timing waarop nieuwe versies geïnstalleerd (moeten) worden een voorbeeld van een afhankelijkheid die de aanbestedende organisatie kan krijgen van de leverancier. Majeure versiewijzigingen kunnen een grote impact hebben op de planning van een organisatie. Propriëtaire software biedt doorgaans niet de mogelijkheid om updates van de nieuwe versie geschikt te maken voor de oude versie en een versiewijziging zo uit te stellen. Ook deze wens tot wendbaarheid kan een rol spelen in de gunningscriteria.

[72] Soms wel pas na een periode, zodat gebruikers de kans hebben gehad om een kwetsbaarheid te herstellen met een update.
[73] Gemiddeld zo'n 75%, zie: Playbook Publieke Software Aanbesteden v2.0.
[74] Zie het voorbeeld op pagina 4 van Playbook Publieke Software Aanbesteden v2.0.
[75] De auteurswet, en Directive 2009/24 EC zeggen dat aanpassingen ten behoeve van fouten en interoperabiliteit mogelijk zijn, ook zonder toestemming van de maker. Hierover is ook jurisprudentie (https://curia.europa.eu/juris/document/document.jsf?docid=247056&doclang=NL&part=1&occ=first&mode=DOC&pageIndex=0&cid=4329854).

Scenario *geen eisen* - maatwerk basissysteem en onderhoud



Ook dit onderdeel moet - in dit scenario waarin we geen bijzondere eisen stellen aan het basissysteem in termen van rechten - wel onderdeel zijn van één en dezelfde aanbesteding. We weten niet zeker of uit de aanbesteding een basissysteem komt dat voldoende vrijheden biedt om van aanpassingen daaraan een apart werkpakket te kunnen maken.

Bij maatwerk aan een basissysteem kunnen we bijvoorbeeld denken aan een nieuw plug-in punt, zoals in het voorbeeld van een database ontsluiten via internet. Er komt dan geheel nieuwe functionaliteit beschikbaar in het basissysteem, die ook niet te scheiden is van het basissysteem.

Korting op maatwerk
In dit scenario is de rol van marktonderzoek niet gericht op onafhankelijkheid, maar op de potentiële commerciele betekenis van het maatwerk voor de leverancier

In een ideale situatie is maatwerk geen maatwerk, maar een uitbreiding die onderdeel wordt van het basissysteem. Als dit het geval is, dan heeft dit maatwerk later ook geen eigen onderhoud nodig. In het voorbeeld van een nieuw plug-in punt is dat een waarschijnlijk scenario. Dit is functionaliteit die niet voor slechts één organisatie bruikbaar is, maar een generiek bruikbare functionaliteit. De leverancier zal in zo'n geval blij zijn om in een betaalde opdracht de basisfunctionaliteit te kunnen uitbreiden. Als de aanbestedende partij voldoende inzicht heeft in de functionele vraag die hij stelt, kan dit aanleiding zijn om bijvoorbeeld de rechten op deze code te delen met de leverancier en zo een korting te bedingen. Tegelijk heeft ook de aanbestedende partij een groot belang bij het opnemen van maatwerk in het basissysteem. Wanneer maatwerk op termijn onderdeel wordt van het basissysteem, weet hij dat er bij nieuwe versies van de software rekening gehouden wordt met de aanpassingen die hij verzoekt. Bij nieuwe versies zal dit maatwerk niet of veel minder terugkomen in de vorm van onderhoud aan dit maatwerk.

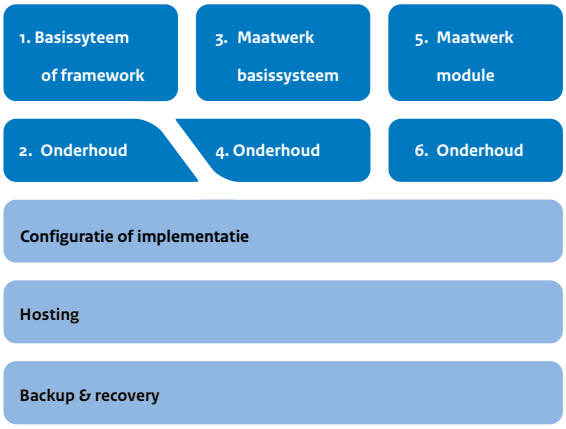
Let op: Uitbreidingen aan het basissysteem kunnen dus aanleiding zijn voor een onderhandeling. Wanneer het basissysteem zelf onder private controle staat, is een financiële korting of kickback vaak het enige interessante onderhandelingsresultaat.

Uiteraard is er zelden sprake van een ideale situatie. Doorgaans kan maatwerk slechts voor een deel gegeneraliseerd worden. De rechten op maatwerk op een basissysteem zijn relatief waardeloos zonder de rechten op het basissysteem of framework zelf. Omdat maatwerk aan een basissysteem zelf ook altijd het wijzigen van de broncode van het basissysteem impliceert. Zelf onderhoud uitvoeren, kan alleen wanneer niet enkel het eigendom van de broncode van het maatwerk wordt verkregen, maar ook de rechten om het basissysteem zelf te mogen gebruiken, begrijpen én wijzigen.

In deze situatie - waarin geen bijzondere *eisen* gesteld worden aan de rechten - kan er dus altijd een grote afhankelijkheid ontstaan. Als geen van de aanbieders voldoende ruime rechten aanbiedt op het basissysteem, dan is volledig eigendom van de maatwerkcode onvoldoende om zelf onderhoud te kunnen plegen of het onderhoud aan de maatwerkcode over te dragen.

In dit geval is het dus bij de gunningscriteria essentieel dat eventuele ruime rechten op de maatwerkcode ***samengaan*** met voldoende ruime rechten op het basissysteem of framework. Het is dus goed om te testen of het mogelijk is om hoger te scoren met beperkte rechten op het basissysteem en ruime rechten of volledige rechten op de maatwerk software. Bij beperkte rechten op het basissysteem is hier altijd sprake van volledige lock-in en kan alleen de leverancier het maatwerk onderhouden. Zonder eisen aan de rechten op het basissysteem lijkt dit ook niet te voorkomen. Daarom vallen onderhoud en maatwerk samen.

Scenario *geen eisen* - maatwerkmodule en onderhoud



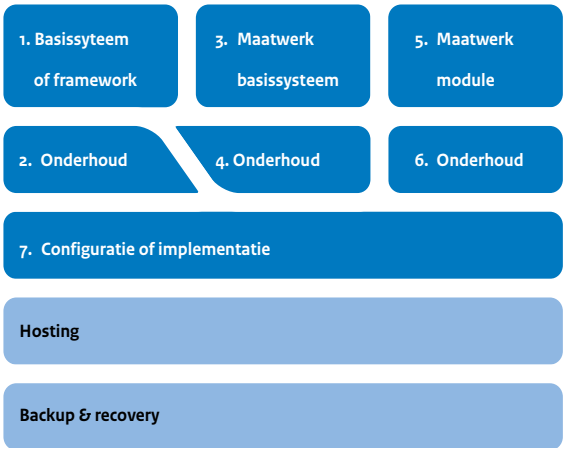
Anders dan in de situatie hiervoor is een maatwerkmodule te onderhouden zonder de code van het basissysteem te kennen of wijzigen. In principe is een maatwerkmodule ook te ontwikkelen zonder de code van het basissysteem te kennen of te wijzigen. Of het praktisch ook mogelijk is om een maatwerkmodule zelf te onderhouden of door een andere partij te laten maken of onderhouden is afhankelijk van de voorwaarden in het basissysteem die beschreven zijn bij het eerste voorbeeld. Uiteraard moet de basissoftware überhaupt een plug-in punt bevatten, moet deze ***interface*** goed genoeg gedocumenteerd zijn, moet deze documentatie ter beschikking gesteld mogen worden aan andere leveranciers en de interface stabiel zijn. Hoewel dit in principe mogelijk is, schrijft PIANOo dat het raadzaam is om afspraken te maken over het afstaan van de broncode indien de koper de software zelf wil (laten) onderhouden[76]. Het derde scenario hieronder, is het scenario waarin dit als eis gesteld wordt.

Als in deze situatie een maatwerkmodule in een apart werkpakket onafhankelijk wordt verworven en er blijkt achteraf toch inzicht nodig in de broncode of aanpassing van de broncode nodig, dan vraagt dit medewerking en meerwerk van de leverancier van het basissysteem. Als de maatwerkmodule als onderdeel van één aanbesteding met het basissysteem wordt verworven, dan kan in elk geval juridisch op resultaat afgerekend worden. Dus hoewel dit een onderdeel is dat in principe onafhankelijk verworven en onderhouden kan worden, leidt bundeling hier tot meer zekerheid. Dit blijft het geval wanneer ruime gebruiksrechten hoger gewaardeerd worden. Het is immers niet zeker dat de uitkomst zal zijn dat ruimere gebruiksrechten verworven worden.

Op dezelfde manier is er in principe iets voor te zeggen om de voorwaarden om het onderhoud van de maatwerkmodule overdraagbaar te maken hoger te waarderen. Tegelijkertijd is onzeker of die voorwaarden ook de uitkomst zullen zijn van de verwerving. En in het geval niet voldaan is aan die voorwaarden, wil je ook het onderhoud gecombineerd hebben in één aanbesteding.

Controle over het basissysteem heeft in zekere zin een virale werking in de context van een aanbesteding. Er blijft altijd sprake van een grote afhankelijkheid van de leverancier van het basissysteem. Dat komt omdat elke wijziging daarin in de vorm van updates of nieuwe versies direct tot noodzakelijke aanpassingen kan leiden in de maatwerkmodule.

Scenario *geen eisen* - configuratie en integratie



Basissystemen bevatten allerlei generieke elementen die geselecteerd moeten worden of ingesteld moeten worden. In ons voorbeeld van het ontsluiten van een database via internet zal de module gespecificeerd moeten worden van de databases die gebruikt worden. Ook staat software vrijwel nooit op zichzelf. Alle software zal moeten samenwerken met andere software. In ons voorbeeld zal de software moeten samenwerken met een webserver en mogelijk geïntegreerd worden met maatwerkmodules. Zoals gezegd, regelmatig is wat we maatwerksoftware noemen in werkelijkheid niet meer dan een uitgebreide vorm van configuratie en integratie.

Het is in dit scenario goed mogelijk dat een andere partij dan de leverancier van de andere onderdelen alle software configureert en integreert. Wat we in het jargon van businessmodellen ook wel “off the shelf” software noemen, wordt vaak geleverd door een zogenaamde “reseller”. Deze verkoopt dan licenties van een derde partij en configureert en integreert de software met een database

[76] https://www.pianoo.nl/nl/sectoren/ict/ict-categorieen/maatwerksoftware.

en eventuele andere benodigde software op een besturingssysteem. Het zijn dan de configuratiebestanden en de aanpassingen aan templates[77] die je als koper zeker wilt stellen. Want dat is wat de software bruikbaar maakt in jouw context, in jouw organisatie. Waar de grens ligt tussen een 'instelling' of 'configuratie' die hoort bij het gebruik van de software en 'code' waarop je ook andere 'rechten' wilt hebben kan voor discussie vatbaar zijn, maar duidelijk is dat je afspraken wilt maken over het afstaan van deze configuratiebestanden en scripts.

Configuratie en integratie valt onder het gebruik van software en in principe is dit werk volledig te onderscheiden van het maken of aanpassen van software. Daarom komt het ook in de wereld van gesloten software voor dat leveranciers software leveren van een andere partij - "resellers". Zij leveren daar vaak ook maatwerkmodules bij en bieden aan het geheel te configureren en integreren. De standaard is dat de code van maatwerksoftware volledig eigendom wordt van de koper, omdat dit in de verschillende inkoopvoorwaarden bij overheden is vastgelegd. Als precies dit gebeurt - een derde partij levert een maatwerkmodule, die volledig in eigendom komt bij de koper - dan is het denkbaar dat wederom een andere partij de configuratie en integratie doet. Dat heeft voordelen, omdat dan de documentatie getest wordt en een andere partij kennis opdoet van de code.

Echter, aanbieders kunnen een redelijk alternatief voorstellen voor onderdelen van de inkoopvoorwaarden. Het afwijzen van zo'n alternatief moet draagkrachtig gemotiveerd kunnen worden. In deze situatie is vooraf niet zeker of we andere rechten zullen verwerven dan het gebruiksrecht. Want in deze situatie hebben we niet in de strategische visie beargumenteerd waarom we dat vereisen. Het is dus onzeker of we voldoende rechten zullen verwerven om een andere leverancier onafhankelijk te laten werken. Daarom is er toch meer zekerheid als de maker van de maatwerkmodule ook degene is die deze configureert en integreert en dit zal dus onder één aanbesteding moeten vallen en daarmee onder dezelfde leverancier.

Tegelijk willen we een partij die alles configureert en integreert tot een werkend geheel, niet voor de configuratie van de maatwerkmodules alleen. En leveranciers van gesloten code van anderen hebben uiteindelijk een afhankelijkheidsrelatie. Het oplossen van problemen in het basissysteem kan uiteindelijk alleen de maker. Dus uiteindelijk moeten al deze afhankelijkheden worden zekergesteld in één aanbesteding. Dus hoewel configuratie en integratie vallen onder het gebruik van software en dit werk in principe volledig te onderscheiden is van het maken en aanpassen van software, is dit werk zeker niet triviaal. Enerzijds omdat dit ook vaak behelst dat er data gemigreerd moet worden. Maar anderzijds omdat er bij meer complexe vragen fouten of problemen kunnen blijken in de basissoftware, of simpelweg aanpassingen in de basissoftware de beste oplossing zijn. Daarom moet ook dit werk min of meer noodzakelijk onder dezelfde aanbesteding gebracht moet worden, zeker als er in deze situatie sprake is van maatwerk.

Scenario 3: vereiste van inzage en distributie

Net zoals in het tweede scenario *Geen eisen* wordt de keuze voor open source of propriëtaire software aan de markt gelaten in dit scenario. Het belangrijkste verschil is uiteraard de eis van inzage en distributie zelf, zoals die beschreven is aan het einde van hoofdstuk 4. Maar hoewel de voorkeur voor open source software tot uitdrukking kan komen in een hogere waardering voor ruimere gebruiksrechten, is het vooraf niet zeker welke rechten zullen worden verworven.

Op één punt verandert de zaak aanzienlijk. Eerder werd beschreven dat wanneer een interface (een plugin punt in het voorbeeld) goed gedocumenteerd is en stabiel, in principe een maatwerkmodule onafhankelijk van de leverancier van de basissoftware ontwikkeld kan worden. De beschikbaarheid van de broncode in dit scenario maakt dat meer reëel[78].

Echter de afhankelijkheid blijft. De controle over de de interface en de stabiliteit van de interface ligt in principe in handen van de leverancier van de basissoftware.

[77] Veel configuratie is al "voorgeconfigureerd". Dat wil zeggen dat er een voorbeeld is gemaakt, vaak met veel commentaar over hoe het voorbeeld aangepast kan worden. Dit noemen we een template.

[78] In principe zegt de auteurswet en Directive 2009/24 EC dat aanpassingen ten behoeve van fouten en interoperabiliteit mogelijk zijn. Hierover is ook jurisprudentie (https://curia.europa.eu/juris/document/document.jsf?docid=247056&doclang=NL&part=1&occ=first&mode=DOC&pageIndex=0&cid=4329854). Een werkwijze zonder broncode (en medewerking van de maker) - via bestudering en/of decompilatie - is echter verre van ideaal. Dat is eerder een noodgreep.

Scenario 4: vereiste van gebruik, begrip, wijziging en distributie

De vereiste van gebruik, begrip, wijziging en distributie zijn dezelfde rechten die door elke open source licentie worden verleend. Ook volledig eigendom omvat alle vier deze rechten. Toch is het belangrijk om te zien dat de vereiste van gebruik, begrip, wijziging en distributie iets anders is dan de vereiste van een open source licentie en ook iets anders dan de vereiste van de overdracht van alle (overdraagbare) intellectuele rechten en dus de verkoop van het volledig eigendom.

Overdracht van eigendom

Wanneer de opdrachtnemer als onderdeel van de verwerving zijn intellectuele eigendomsrechten overdraagt - ofwel verkoopt - dan verkrijgt de opdrachtgever deze rechten volledig[79]. Er is dan geen sprake van een licentie[80]. Er zijn twee dingen die een eigenaar kan doen, die de eigenaar van een licentie niet[81] kan doen. Het eerste is de software onder willekeurig welke andere licentie uitbrengen, "herlicenseren". Het andere is het eigendom overdragen.

Een eigenaar kan via (aanbestedings-) contracten code in bruikleen geven wanneer andere partijen hieraan een bijdrage leveren. Eigendom maakt het dus net als open source licenties mogelijk om meerdere partijen gelijktijdig aan software te laten werken die een permanent onderdeel van de organisatie is en kan blijven.

Rechten onder een licentie

Rechten onder een licentie betekent dat de koper toestemming krijgt om iets te doen onder voorwaarden, al zijn die voorwaarden nog zo minimaal. Dat betekent bijvoorbeeld dat hij als niet-auteursrechthebbende de licentie niet eenzijdig kan veranderen of naast de licentie contracten kan sluiten met een leverancier die de (open source) licentie niet toestaat.

Waarom is dit verschil van belang bij de inkoop?

De vier eisen in dit scenario kunnen dus op twee heel verschillende manier worden ingevuld met verschillende consequenties voor de aanbesteding. Op de eerste plaats bepalen de eisen die gesteld worden in een aanbesteding bepalen wat een leverancier kan leveren en tegen welke kosten.

A) Wanneer volledig eigendom vereist wordt

Wanneer de koper volledig eigendom eist, dan kan de leverancier dit alleen leveren, wanneer die volledig eigendom heeft. Een licentie geeft toestemming om iets te doen en doorgaans onder voorwaarden, al zijn die voorwaarden soms minimaal. Dat betekent bijvoorbeeld dat een leverancier niet slechts een toevoeging kan leveren aan bestaande software die hij niet in bezit heeft. Dit maakt het product duurder[82]. Dit is onnodig, in dit scenario waarin de koper alleen toestemming nodig heeft tot gebruik, begrip, wijziging en distributie.

- Eigendom vereisen maakt software onnodig duur

Zeker wanneer het ook mogelijk zou zijn om voort te bouwen op code die beproefd is en onderhouden wordt door meerdere partijen, is dit ook een risico voor de kwaliteit de software die de koper verkrijgt.

Wie volledig eigendom vereist, verkrijgt - bezien vanuit de koper - de mogelijkheden die open source software ook biedt *in principe*. Want anders dan in het eerste scenario *Open source vereist*, is de frictieloze overdracht die gestandaardiseerde open source licenties bieden er niet. Ook het garanderen van de integriteit van de code verloopt niet via de licentie. Dat wil zeggen dat iedereen die een bijdrage levert - externe bedrijven, eigen of gedetacheerde medewerkers - ook toestemming heeft en geeft[83]. Dus zal de organisatie een procedure en registratie moeten instellen hiervoor.

- Eigendom vereist registratie van bijdragen en bijbehorende overeenkomsten

[79] Als de eigenaar (diverse) externe partijen laat werken aan de code en/of ook eigen of gedetacheerd personeel, dan is het van belang om een procedure en registratie in te stellen om de integriteit van de bijdragen te garanderen. Dat is nodig enerzijds omdat dit niet direct via een open source licentie verloopt. Anderzijds is dit van belang om later alsnog met vertrouwen te kunnen voldoen aan de beleidslijn waarin software die de overheid ontwikkelt of laat ontwikkelen door de organisatie zelf onder een open source licentie wordt uitgebracht.

[80] Uiteraard zal de eigenaar - conform de beleidslijn "Open, tenzij" - moeten beargumenteren welke gegronde redenen er zijn om specifieke delen van de software in eigendom te nemen en de delen waarvoor dit niet geldt zelf moeten vrijgeven onder een open source licentie. Want van deze beleidslijn kan worden afgeweken als er gegronde redenen zijn te vinden in belangen van nationale of openbare veiligheid. Zie de uitzonderingen op pagina 2 in de overwegingen of pagina 4 van het afwegingskader ict opdrachten.

[81] Het is mogelijk om - als beheerder van de code - met behulp van een zogenaamd "contributor (license) agreement" alsnog de mogelijkheid te scheppen om de software te herlicenceren. Een CLA is een contract waarbij een persoon die een bijdrage levert aan een OSS project, de beheerder van dat project bepaalde rechten geeft. Er is discussie mogelijk over of en wanneer dat nodig of wenselijk is.

[82] Niet voor niets lezen we op de website van Pianoo dat in zulke gevallen de inschrijvingsprijs waarschijnlijk erg hoog is. https://www.pianoo.nl/nl/sectoren/ict/ict-categorieen/maatwerksoftware.

[83] Tijdens de interviews voorafgaand aan Playbook Publieke Software Aanbesteden v2.0 bleek dat dit iets is dat overheden bij code in eigen bezit niet altijd goed op orde hebben.

Een koper kan uiteraard ook zelf de code onder een open source licentie uitbrengen - zoals de beleidsbrief "Open, tenzij" van overheden vraagt.

B) Wanneer volledig eigendom vereist wordt op toevoegingen

Het is ook mogelijk om het volledig eigendom te vereisen op de resultaten van verrichte diensten voor zover dat toevoegingen zijn op bestaande software. Dit is de strategie die algemene inkoopvoorwaarden als ARBIT en ARVODI hanteren. Die stellen dat voor zover de prestatie (mede) tot stand komt op basis van bestaande software moet de leverancier een gebruikslicentie leveren[84].

Wanneer expliciet de mogelijkheid wordt geboden om software van derden te gebruiken, die dan onder licentie - een gebruikslicentie of ruimere licentie - geleverd wordt, maar volledig eigendom op de toevoegingen van de leverancier, sluit dit het voortbouwen op software onder beschermende licenties in een aantal gevallen uit. Dat komt omdat deze licenties vereisen dat deze wijzigingen met *dezelfde* rechten geleverd moeten worden wanneer ze worden overgedragen.

Dat betekent dat wanneer maatwerk geleverd wordt aan een basissysteem, *het geheel* met dezelfde rechten geleverd moet worden. Dus niet overgedragen kan worden, maar onder een (of die) open source licentie geleverd moet worden. Uiteraard is het mogelijk om slechts de wijzigingen in over te dragen en staat ook niks in de weg om een externe partij basissysteem en wijzigingen in de eigen organisatie te laten integreren. Maar we komen hier op het terrein van de kleine lettertjes. Het vereisen van wat in dit scenario nodig is, het recht op gebruik, begrip, aanpassing en distributie, is een stuk eenvoudiger. Tegelijk vraagt dit - zoals nog altijd bij open source software - om een uitzondering op de inkoopvoorwaarden.

C) Wanneer de koper het recht op gebruik, inzage, wijziging en distributie vraagt

Wanneer de koper alleen de rechten vereist, die nodig zijn om (later) ook andere bedrijven te laten werken aan de code die de koper zelf in gebruik neemt, maakt dit het maximaal mogelijk voor leveranciers om voort te bouwen op bestaande code. Leveranciers met veel kennis van wat er reeds beschikbaar is aan software, kan soms met een (relatief) kleine toevoeging software leveren. Dat kan veel schelen in de kosten.

Net als bij volledig eigendom maakt het vereisen van het recht van gebruik, inzage, wijziging en distributie samenwerking en overdracht mogelijk *via (contracten met) de opdrachtgever*.

D) Wanneer de koper een open source licentie vereist

Tenslotte - maar dit is het eerste scenario - is het uiteraard mogelijk om een open source licentie te vereisen. In dat geval kan de software overgedragen worden *via de licentie*. Dat wil zeggen: gebruik van de rechten in de licentie, is acceptatie van de licentie en daarmee van de voorwaarden.

[84] Sinds enkele jaren is dit de strategie die inkoopvoorwaarden als ARVODI en ARBIT hanteren. ARVODI vereist eigendom op de resultaten (24.1) en voor zover die resultaten (mede) tot stand komen met bestaande intellectuele eigendomsrechten niet in bezit van leverancier (24.3), dan moet leverancier daarop een niet-exclusieve, niet-opzegbare gebruiksrecht leveren voor onbepaalde duur. ARBIT volgt in artikel 8 eenzelfde soort strategie.

8 Na de aanbesteding - governance van de software

Aanbesteding van software zou geen eenmalige gebeurtenis moeten zijn. Maar juist wanneer een aanbesteding als eenmalige gebeurtenis de start van een softwareproject inluidt voor een organisatie of groep van organisaties, is het van belang te bedenken hoe er met de software en de markt van softwareleveranciers wordt omgegaan. Een visie daarop kan het mogelijk maken om niet alleen in de initiële aanbesteding te werken met kleinere eventueel parallelle opdrachten, maar vooral ook daarna. Daarbij vragen de volgende vragen aandacht.

Wat is de strategie voor het beheer van de software?

De eerste vraag daarbij is wie de eigenaar wordt. Zelf eigenaar zijn betekent zelf het beheer organiseren. Dat betekent dus ook procedures voor het documenteren van dat eigenaarschap. Uiteraard kan ook een daartoe uitgeruste organisatie eigenaar worden en het beheer daarvan uitvoeren. Het eigenaarschap van elk stuk originele code kan bij de leverancier gelaten worden van dat stuk code, de overdracht onder een open source licentie en het beheer - eventueel van een eigen distributie (fork) - kan aan wisselende bedrijven uitbesteed worden. Het eigenaarschap kan bij de originele maker teruggelegd worden, bijvoorbeeld wanneer het de bedoeling is dat de software publiek beschikbaar is en die leverancier een geloofwaardige strategie heeft voor het exploiteren van het product gebaseerd op open source licenties.

Welke licentie kies ik?

Er kunnen redenen zijn om een specifieke licentie te verkiezen. Het kan van belang geacht worden om te verzekeren dat software die met publieke middelen is gemaakt niet onder door een private partij geëxploiteerd kan worden met een kleine toevoeging. Of het kan wenselijk zijn dat dit juist wèl kan, zodat bedrijven hier - wellicht ook buiten een overheidsmarkt - een verdienmodel op basis van intellectueel eigendomsrecht kunnen baseren. Apple is er groot mee geworden. De inbedding in het Europees recht kan van groot belang geacht worden, wellicht mede vanwege (potentiële) Europese samenwerking tussen soortgelijke overheidsorganisaties. Wat de keuze ook is, het is verstandig om hierin keuzes te maken, zodat overzicht en routines kunnen ontstaan.

Ontwikkel een geïntegreerde aanbestedings- en beheerstrategie

Uiteindelijk moeten alle vragen die in dit document spelen veel minder strategische vragen worden, maar een strategie die is uitgekristalliseerd in werkroutines. Routines waarin juridische aspecten een praktische implementatie hebben gekregen, zoals hoe te voldoen aan de eisen van de licentie of hoe de integriteit van de code wordt zeker gesteld.

9 Literatuur

Black Duck by Synopsis, (2022), *Open Source Security and Risk Analysis (OSSRA) Rapportage 2021*. (https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html)

Ministerie van BZK, (2020), *Kamerbrief over vrijgeven broncode overheidssoftware* (https://www.rijksoverheid.nl/documenten/kamerstukken/2020/04/17/kamerbrief-inzake-vrijgeven-broncode-overheidssoftware)

Paapst, M., 2013, *Barrières en doorwerking: Een onderzoek naar de invloed van het open source en open standaarden beleid op de Nederlandse aanbestedingspraktijk.* PhD Rijksuniversiteit Groningen. (https://www.rug.nl/research/portal/nl/publications/barrieres-en-doorwerking(cb68c60a-dce2-4451-8b43-7c73429e8da8).html)

Plantinga, E., Groenen, J., Bouwkamp, H. en Gerrits, A., (2018_, *Winst op (tien) punten*, iBestuur. (https://ibestuur.nl/nieuws/winst-op-tien-punten)

Raymond E.S., (1999), *The Cathedral and the Bazaar: Musings on Open Source and Linux by an Accidental Revolutionary*, O' Reilly.

Schryen, G., 2011, *Is open source security a myth?* Communications of the ACM (https://doi.org/10.1145/1941487.1941516)

Stichting Kafkabrigade (2021), *Playbook Publieke Software Aanbesteden v2.0*, rapport in opdracht van het Ministerie van Binnenlandse Zaken en Koninkrijksrelaties, (http://kafkabrigade.nl/playbook of https://www.rijksoverheid.nl/documenten/brochures/2021/01/05/playbook-publieke-software-aanbesteden)

Van der Heijden, Jurgen., (2001), *Een filosofie van behoorlijk bestuur. Een verklaring voor de juridische en de maatschappelijke functie van de beginselen van behoorlijk bestuur*. (PhD. Amsterdam UvA) W.E.J. Tjeenk Willink, Deventer. (Doi: https://hdl.handle.net/11245/1.254553)

Widlak, A.C., (2021a), *Volwassen Digitale Overheid*, Boombestuurskunde, Den Haag.

Widlak, A.C., (2021b), *Is open source goed voor de veiligheid?*, iBestuur. (https://ibestuur.nl/podium/is-open-source-goed-voor-de-veiligheid)

Widlak, A.C., (2022), *De beste bureaucratie ter wereld*, iBestuur. (https://ibestuur.nl/podium/de-beste-bureaucratie-ter-wereld)

Wolswinkel, J., (2020), *Willekeur of algoritme? Laveren tussen analoog en digitaal bestuursrecht*. Tilburg University.

10 Credits

Deze handreiking is geschreven door Arjan Widlak van Stichting Kafkabrigade. Bij de totstandkoming hebben allerlei mensen meegewerkt aan de hand van interviews en commentaren. Niet alle namen zijn bekend en niet iedereen wilde vermeld worden, maar onder meer deze mensen hebben bijdragen aan dit document:

Koos Steenbergen (ministerie van Binnenlandse Zaken en Koninkrijksrelaties), Mathieu Paapst (Rijksuniversiteit Groningen), Daan Moerman (UBR), Rolf Zeldenrust (Pianoo), Henk Jan Siersema (Pianoo), Rik Swertz (UBR), Fredo Schotanus (Universiteit Utrecht, Significant), Jacques van Berkel (Advocaat), Walter van Holst (Hooghiemstra en partners), Maurice Hendriks (Gemeente Amsterdam), Jacco Brouwer (VNG Realisatie), Gerbrand van Dieijen (Marktplaats), Jani Kylmäaho (National Land Survey of Finland), Arnoud Engelfriet (ICTRecht, Lynn Legal, JuriBox)

11. Licentie

Deze brochure is opgesteld in opdracht van het
Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

Mei 2022