Ministerie van Volksgezondheid,
Welzijn en Sport

Open source ambitieladder voor opensourcewerken in projecten

Maurice Hendriks
In samenwerking met alle bijdragers

Gepubliceerd op 17 december 2025

Ministerie van Volksgezondheid,
Welzijn en Sport

Inhoud

Introductie

Deze ambitieladder maakt het mogelijk om na te denken over de ambitie van opensourcewerken voor projectteams en daar ook concrete eisen aan te stellen. Het maakt voor deze ambitieladder niet uit waar en wie die projectteams zijn. Of dat nu interne teams zijn die werken aan software of teams bij een leverancier die maatwerkontwikkeling uitvoeren.

Dit document is zorgvuldig opgebouwd met zoveel mogelijk samenhang tussen de ambities en de eisen die er per ambitie worden gesteld. Daarmee vormt dit document een handige leidraad voor het opensourcewerken. Elke ambitie is zo geformuleerd dat er ook een intentie uit spreekt. Dat wil zeggen dat de bedoeling uit de verwoording van de ambitie moet blijken. Mochten de eisen die per ambitie gesteld worden op een of andere manier vragen oproepen, dan moet de ambitie als intentie voor zichzelf kunnen spreken. Waardoor je alsnog het gesprek kan voeren over hoe je in jouw situatie het best aan die intentie kan voldoen.

Dit document verwoord dus vooral geen wetmatigheid. Het belangrijkste is dat er concrete afspraken worden gemaakt over wat precies onder opensourcewerken wordt verstaan. Zodat er achteraf geen verkeerde verwachtingen of aannames ontstaan. Het staat dus iedereen vrij creatief om te gaan met alle informatie in dit document. Door verschillende ambitieniveaus door elkaar heen te gebruiken of door eisen te stellen die niet bij een bepaald ambitieniveau horen. Wat voor jouw situatie werkt.

🔍 Gebruik van Generatieve AI

Bij dit document is op drie manieren gebruikgemaakt van Generatieve AI (GenAI):

- GenAI heeft de teksten gereviewd op flow, taal en spelling.
- GenAI is gevraagd aanvullende suggesties te doen op alle teksten.
- GenAI is op specifieke punten gevraagd om te sparren.

Voordat GenAI om input wordt gevraagd, worden alle teksten eerst door de hoofdauteur of bijdragers zelf geschreven. Zo wordt voorkomen dat hier informatie wordt gedeeld waarvan de (hoofd)auteur(s) zelf geen achtergrondkennis heeft/hebben en waarvan onduidelijk is wat de doorwerking is. Door zelf eerst teksten te schrijven, wordt het ook voor GenAI duidelijker wat precies de bedoeling is van een specifiek stuk tekst.

Verder wordt aan dit document in openbaarheid gewerkt. De hier benodigde kennis rond open source, wetgeving en overheidsbeleid is doorgaans niet vertrouwelijk van aard en vrij te gebruiken.

Als er met GenAI is gespard over specifieke teksten, dan is daar in de Rationale een toelichting bij gegeven.

Hierdoor wordt het gebruik van GenAI verantwoord geacht en als waardevolle aanvulling gezien om tot een zo bruikbaar mogelijk instrument te komen.

💡 Standaard voor publieke code

Voor dit document zijn verschillende bronnen ter inspiratie gebruikt. Eén daarvan is de Standaard voor Publieke Code. Deze standaard heeft alleen drie nadelen:

1. De standaard gaat uit van perfectie. Dat wil zeggen dat het voor ontwikkelteams niet duidelijk is welke minimale eisen je als ontwikkelteam moet hanteren om bepaalde doelen te bereiken.
2. De standaard maakt het niet praktisch genoeg wat je moet doen om aan een bepaalde eis te voldoen. Daarvoor zijn de eisen te strategisch van aard.
3. De standaard sluit niet naadloos aan bij de communicatielijn over open source vanuit BZK.

💡 Maak iemand verantwoordelijk voor het publicatieproces

Deze ambitieladder neemt je mee in alle aspecten die een rol spelen bij opensourcewerken in projecten. Het is voor partijen voor wie opensourcewerken nieuw is allemaal overweldigend. Het is in dat geval aan te raden iemand in het team verantwoordelijk te maken voor de naleving voor alle van toepassing zijnde punten. Wanneer er niet volledig in openbaarheid wordt ontwikkeld, dan is het sowieso raadzaam extra alert te zijn op de publicatiemomenten.

Het is goed om te realiseren dat - afhankelijk van de dynamiek die in openbaarheid open source werken met zich meebrengt - dit een zware verantwoordelijkheid kan zijn. Het betekent namelijk dat je (periodiek) in contact zal moeten staan met mogelijke Woo-verzoekers, juristen, communicatie, het projectteam en management. Bij kleinere minder politiek gevoelige onderwerpen kan dit meegenomen worden door de productowner. Ook als de productowner al ervaring heeft met deze trajecten. Anders is het aan te raden deze verantwoordelijkheid bij een projectleider neer te leggen.

❗ Het delen van broncode

De basis van het opensourcewerken is dat broncode wordt gepubliceerd in een voor de betreffende programmeertaal gangbaar formaat, op platformen als GitHub of GitLab. Anders is er ook niet transparant samen te werken op de broncode.

Het wil vanuit een Woo-verzoek namelijk ook voorkomen dat broncode wordt uitgeprint, maar is sterk af te raden wegens de slechte verwerkbaarheid van zulke broncode.

Heb je eenmaal broncode openbaar gemaakt, zorg dan dat deze geïndexeerd is door in ieder geval het Woo-platform van je eigen organisatie en in de catalogus van https://developer.overheid.nl.

Voordelen van opensourcewerken

Dit document focust zich op de vier voordelen van open source zoals gecommuniceerd door het programma Opensourcewerken. Dat wil niet zeggen dat elk voordeel voor elke situatie even belangrijk is of op dezelfde manier nagestreefd zou moeten worden. Per voordeel zijn dus ambities of opties te formuleren. Deze kan je vrij combineren om je eigen variant samen te

stellen.

1. **Transparantie en Vertrouwen**
   Iedereen die dat wil, kan zien hoe de software werkt en in elkaar zit. Dit leidt tot vertrouwen in digitale diensten van de overheid en haar partners.
2. **Samenwerking en Innovatie**
   Verschillende overheidsinstellingen, ondernemers en inwoners kunnen samenwerken aan software, en van elkaar leren. Ontwikkelaars kunnen gebruikmaken van elkaars ideeën, en samen sneller innoveren.
3. **Veiligheid en Betrouwbaarheid**
   Iedereen kan fouten of risico's ontdekken en onderzoeken. Dit vergroot de veiligheid en betrouwbaarheid van bijvoorbeeld apps en inlogmethodes.
4. **Efficiëntie en Onafhankelijkheid**
   Doordat open software herbruikbaar is, kan opensourcewerken innovatie en ontwikkelprocessen versnellen. Overheden kunnen bijvoorbeeld software van elkaar gebruiken. Dit voorkomt dubbel werk en onnodige kosten. Ook is de overheid niet afhankelijk van één softwareleverancier.

In zijn algemeenheid is het goed om na te denken over hoe de doelstellingen eruit zien op korte, middellange en lange termijn? Welke samenwerking met stakeholders (leveranciers, partners en/of overige geïnteresseerden) is daarvoor nodig over 1, 5 en 10 jaar?

Overwegingen

Per voordeel staan drie ambitieniveaus geformuleerd, die grofweg het volgende groeimodel volgen:

1. Basishygiëne
2. Samenwerken met anderen
3. Volledig opensourcewerken

Elk ambitieniveau borduurt voort op de vorige ambitie en wordt daarmee ambitieuzer. Als een onderdeel in een opvolgende ambitie hetzelfde blijft, dan wordt deze herhaald. Als er een onderdeel bijkomt, er een veranderd of vervalt, dan wordt dit met de rood/groen verduidelijkt. Er wordt bewust gekozen om telkens in herhaling te vallen zodat het makkelijker is direct in te prikken op een hoger ambitieniveau, zonder eerst alle eisen bij elkaar te moeten sprokkelen.

Transparantie & Vertrouwen

Ambitie 1. Het eenmalig open source publiceren van alle broncode na afronden van het project

- Alle broncode en documentatie wordt na afronding van het project openbaar gedeeld.
  - **Licenties**
    Onderstaande eisen volgen de REUSE standaard voor het correct communiceren van open source licenties en auteursrecht.
    - ☐ De volledige Engelse licentietekst van de open source licentie die geldig is op de hele repository wordt in een plat tekstformaat in de root van de repository gezet. Dit bestand krijgt de naam `LICENSE` (met of zonder txt of md extensie).
    - ☐ Alle andere gebruikte open source licenties worden in een plat tekstformaat in de LICENSE folder gezet. Elke licentiebestand krijgt de naam van de SPDX-identifier van de betreffende licentie. In geval van de European Union Public License versie 1.2 of hoger is dit `EUPL-1.2-or-later.txt`. Van elke licentie wordt de volledige Engelse tekst in het betreffende bestand opgenomen.
    - ☐ In alle broncode is een koptekst toegevoegd met daarin volgens de SPDX confentie:
      - ☐ De van toepassing zijnde open source licentie.
      - ☐ Copyright van de organisaties of individuen door wie het specifieke bestand is ontwikkeld.
    - ☐ Wanneer alle broncode bestanden van een (sub)module onder dezelfde licentie vallen en dit een afwijkende licentie is dan van het hoofdproject, dan wordt een aparte `LICENSE` (met of zonder txt of md extensie) in (sub)module map gezet. Zo wordt direct duidelijk dat een (sub)module losstaand onder die licentie gebruikt kan worden.
    - ☐ Middels REUSE worden bovenstaande eisen geautomatiseerd getest op compliance.
    - ☐ Met een REUSE tag wordt aangeven of de repository daadwerkelijk compliant is.
  - **Beschrijving**
    - ☐ Er wordt een `README.md` in de root van de repository gezet met daarin in ieder geval:
      - ☐ Namens welke organisatie de publicatie is gedaan.
      - ☐ Dat het om een eenmalige publicatie gaat.

    - ☐ Of en hoe iemand contact kan opnemen voor het stellen van vragen.
    - ☐ Technische documentatie over wat de broncode doet en hoe deze gebruikt kan worden of een verwijziging naar de technische documentatie.
- **Documentatie**
    - ☐ Alle documenten in de repository zijn:
        - ☐ In een platte tekst formaat opgesteld, bij voorkeur Markdown.
        - ☐ Of wanneer dit niet mogelijk is in een open bestandsformaat.
    - ☐ Van alle documentatie zijn ook de bronbestanden beschikbaar.
    - ☐ Het is inzichtelijk welke delen van de broncode niet zijn gepubliceerd en om welke reden. Denk daarbij aan een van de uitzonderingsgronden uit de Wet open overheid.
- **Besluitvorming**
    - ☐ Alle besluitvorming is geformaliseerd in documenten, denk aan architectuur, functionele en technische beschrijvingen en kaders.

Ambitie 2. Op vaste momenten open source publiceren van de broncode

- Alle broncode en documentatie worden op afgesproken intervallen of op sleutelmomenten, onder resp. de [open source licentie] en [documentatie licentie], na afronding van de opdracht gepubliceerd en de repository wordt actief beheerd
    - **Licenties**
      Onderstaande eisen volgen de REUSE standaard voor het correct communiceren van open source licenties en auteursrecht.
        - ☐ De volledige Engelse licentietekst van de open source licentie die geldig is op de hele repository wordt in een plat tekstformaat in de root van de repository gezet. Dit bestand krijgt de naam `LICENSE` (met of zonder txt of md extensie).
        - ☐ Alle andere gebruikte open source licenties worden in een plat tekstformaat in de LICENSE folder gezet. Elke licentiebestand krijgt de naam van de SPDX-identifier van de betreffende licentie. In geval van de European Union Public License versie 1.2 of hoger is dit `EUPL-1.2-or-later.txt`. Van elke licentie wordt de volledige Engelse tekst in het betreffende bestand opgenomen.
        - ☐ In alle broncode is een koptekst toegevoegd met daarin volgens de SPDX confentie:

  - ☐ De van toepassing zijnde open source licentie.
  - ☐ Copyright van de organisaties of individuen door wie het specifieke bestand is ontwikkeld.
- ☐ Wanneer alle broncode bestanden van een (sub)module onder dezelfde licentie vallen en dit een afwijkende licentie is dan van het hoofdproject, dan wordt een aparte `LICENSE` (met of zonder txt of md extensie) in (sub)module map gezet. Zo wordt direct duidelijk dat een (sub)module losstaand onder die licentie gebruikt kan worden.
- ☐ Middels REUSE worden bovenstaande eisen geautomatiseerd getest op compliance.
- ☐ Met een REUSE tag wordt aangeven of de repository daadwerkelijk compliant is.

– **Beschrijving**

- ☐ Er wordt een `README.md` in de root van de repository gezet met daarin in ieder geval:
  - ☐ Namens welke organisatie de publicatie is gedaan.
  - ☐ Dat het om een eenmalige publicatie gaat.
  - ☐ In welk interval gepubliceerd wordt.
  - ☐ Of en hoe iemand contact kan opnemen voor het stellen van vragen.
  - ☐ Technische documentatie over wat de broncode doet en hoe deze gebruikt kan worden of een verwijziging naar de technische documentatie.
  - ☐ Wie de beheerder van de repository is.
  - ☐ Of en hoe iemand contact kan opnemen met de beheerder.

– **Documentatie**

- ☐ Alle documenten in de repository zijn:
  - ☐ In een platte tekst formaat opgesteld, bij voorkeur Markdown.
  - ☐ Of wanneer dit niet mogelijk is in een open bestandsformaat.
- ☐ Van alle documentatie zijn ook de bronbestanden beschikbaar.
- ☐ Het is inzichtelijk welke delen van de broncode niet zijn gepubliceerd en om welke reden. Denk daarbij aan een van de uitzonderingsgronden uit de Wet open overheid.

– **Besluitvorming**

- ☐ Alle besluitvorming is geformaliseerd in documenten, denk aan architectuur, functionele en technische beschrijvingen en kaders.

- In de documentatie wordt duidelijk gemaakt wat de verschillen zijn tussen twee gepubliceerde versies.
  - **Versiebeheer**
    - ☐ Er wordt gebruik gemaakt van semantische versienummering voor nieuwe software versies.
    - ☐ Er wordt gebruik gemaakt van YYYY.MM(.DD) versienummer in geval van standaarden en specificaties.
    - ☐ Versies worden aangemaakt door gebruik te maken van git tags.
    - ☐ Er wordt in een change log woordelijk beschreven wat de verschillen zijn tussen de huidige en vorige versie.
    - ☐ Wanneer issues zijn opgelost of merge/pull-requests zijn geaccepteerd, dan worden deze opgesomd in de changelog met een verwijziging naar het issue nummer.
    - ☐ De changelog bevat een apart kopje waarin alle verbeteringen op de documentatie staat beschreven.
  - **Beschrijving**
    - ☐ Als de nieuwe versie van de broncode niet 1-op-1 op dezelfde manier te gebruiken is als de vorige versie, dan wordt duidelijk gemaakt wat de gebruiker moet doen om de nieuwe versie te kunnen gebruiken.

Ambitie 3. Het volledig openbaar open source ontwikkeling van de broncode

- Alle onder de overeenkomst ontwikkelde broncode en documentatie worden ontwikkeld op afgesproken intervallen of op sleutelmomenten in openbaarheid, onder resp. de [open source licentie] en [documentatie licentie] en de repository wordt actief beheerd.
  - **Licenties**
    Onderstaande eisen volgen de REUSE standaard voor het correct communiceren van open source licenties en auteursrecht.
    - ☐ De volledige Engelse licentietekst van de open source licentie die geldig is op de hele repository wordt in een plat tekstformaat in de root van de repository gezet. Dit bestand krijgt de naam `LICENSE` (met of zonder txt of md extensie).
    - ☐ Alle andere gebruikte open source licenties worden in een plat tekstformaat in de LICENSE folder gezet. Elke licentiebestand krijgt de naam van de SPDX-identifier van de betreffende licentie. In geval van de European Union

Public License versie 1.2 of hoger is dit `EUPL-1.2-or-later.txt`. Van elke licentie wordt de volledige Engelse tekst in het betreffende bestand opgenomen.

- ☐ In alle broncode is een koptekst toegevoegd met daarin volgens de SPDX confentie:
  - ☐ De van toepassing zijnde open source licentie.
  - ☐ Copyright van de organisaties of individuen door wie het specifieke bestand is ontwikkeld.
- ☐ Wanneer alle broncode bestanden van een (sub)module onder dezelfde licentie vallen en dit een afwijkende licentie is dan van het hoofdproject, dan wordt een aparte `LICENSE` (met of zonder txt of md extensie) in (sub)module map gezet. Zo wordt direct duidelijk dat een (sub)module losstaand onder die licentie gebruikt kan worden.
- ☐ Middels REUSE worden bovenstaande eisen geautomatiseerd getest op compliance.
- ☐ Met een REUSE tag wordt aangeven of de repository daadwerkelijk compliant is.

– **Beschrijving**

- ☐ Er wordt een `README.md` in de root van de repository gezet met daarin in ieder geval:
  - ☐ Namens welke organisatie de publicatie is gedaan.
  - ☐ In welk interval er gepubliceerd zal worden.
  - ☐ Beschreven dat elke wijziging op de broncode en/of documentatie in openbaarheid te volgen is.
  - ☐ Of en hoe iemand contact kan opnemen voor het stellen van vragen.
  - ☐ Technische documentatie over wat de broncode doet en hoe deze gebruikt kan worden of een verwijziging naar de technische documentatie.
  - ☐ Wie de beheerder van de repository is.
  - ☐ Of en hoe iemand contact kan opnemen met de beheerder.

– **Documentatie**

- ☐ Alle documenten in de repository zijn:
  - ☐ In een platte tekst formaat opgesteld, bij voorkeur Markdown.
  - ☐ Of wanneer dit niet mogelijk is in een open bestandsformaat.
- ☐ Van alle documentatie zijn ook de bronbestanden beschikbaar.
- ☐ Het is inzichtelijk welke delen van de broncode niet zijn gepubliceerd en om welke reden. Denk daarbij aan een van de uitzonderingsgronden uit de Wet open overheid.

  - **Maatregelen**
    - ☐ Het geforceerd bijwerken van historie wordt niet toegestaan op de hoofd commit historie.

- In de documentatie wordt duidelijk gemaakt wat de verschillen zijn tussen twee gepubliceerde versies.
- Zowel de historie en de voortgang van de ontwikkeling volledig te volgen is inclusief de ontwerpkeuzes die tijdens de ontwikkeling zijn gemaakt.

  - **Versiebeheer**
    - ☐ Er wordt gebruik gemaakt van semantische versienummering voor nieuwe software versies.
    - ☐ Er wordt gebruik gemaakt van YYYY.MM(.DD) versienummer in geval van standaarden en specificaties.
    - ☐ Versies worden aangemaakt door gebruik te maken van git tags.
    - ☐ Er wordt in een change log woordelijk beschreven wat de verschillen zijn tussen de huidige en vorige versie.
    - ☐ Wanneer issues zijn opgelost of merge/pull-requests zijn geaccepteerd, dan worden deze opgesomd in de changelog met een verwijziging naar het issue nummer.
    - ☐ De changelog bevat een apart kopje waarin alle verbeteringen op de documentatie staat beschreven.
    - ☐ De integriteit van de hoofd commit historie wordt intact gehouden.
    - ☐ Elke commit bevat een heldere beschrijving van de wijziging of toevoeging die ermee is gedaan.
    - ☐ De scope van elke commit is helder afgebakend. Als voorbeeld: typefouten in documentatie worden niet samen in één commit gezet met een verbetering op een functie in de broncode, als die geen onderlinge samenhang hebben.
  - **Beschrijving**
    - ☐ Als de nieuwe versie van de broncode niet 1-op-1 op dezelfde manier te gebruiken is als de vorige versie, dan wordt duidelijk gemaakt wat de gebruiker moet doen om de nieuwe versie te kunnen gebruiken.
  - **Documentatie**
    - ☐ De broncode bevat technische documentatie met een heldere beschrijving van de werking van elk onderdeel van de broncode.
    - ☐ De broncode wordt zoveel als mogelijk als self-documenting code geschreven.

    - ☐ De technische documentatie wordt opgesteld in een voor de betreffende programmeertalen gangbaar formaat, zodat eventueel met een geautomatiseerde documentatie generator een API beschrijving kan worden gegeven van de broncode. Denk aan Docstrings in geval van Python.
    - ☐ De technische documentatie sluit 1-op-1 aan op de functionele documentatie.
    - ☐ Specifieke ontwerpkeuzes worden onderbouwd met referentie naar idealiter organisatiebrede of anders projectspecifieke architectuurbeschrijvingen en -principes.
    - ☐ Documentatie wordt regelmatig geüpdatet.
  - **Besluitvorming**
    - ☐ Alle besluitvorming is geformaliseerd in documenten, denk aan architectuur, functionele en technische beschrijvingen, kaders etc. óf
    - ☐ Alle discussies worden in openbaarheid
      - ☐ gevoerd als ze mogelijk leiden tot een besluit
      - ☐ gepubliceerd wanneer ze hebben geleid tot een besluit Denk daarbij aan issues, pull- of pull-requests.

Samenwerking & Innovatie

Ambitie 1. Externe bijdragen worden niet behandeld

- De broncode is wel in openbaarheid voor hergebruik gepubliceerd, maar het team staat niet open voor feedback of verbetersuggesties van buiten.
  - **Versiebeheer**
    - ☐ Gepubliceerde versies zijn van elkaar te onderscheiden door consequente versienummering.
  - **Documentatie**
    - ☐ Elke versie van de broncode bevat een Software Bill of Materials (SBoM) waarin in ieder geval de volgende informatie is opgenomen:
      - Naam van het component.
      - Waar de broncode van de component gevonden kan worden.
      - Licentie van het component.
      - Wie de IE-rechten houder is.

☐ Er wordt in de README duidelijk toegelicht dat externe bijdragen niet in behandeling worden genomen.

Ambitie 2. Externe bijdragen worden behandeld maar niet actief gezocht

- De broncode is wel in openbaarheid voor hergebruik gepubliceerd, maar en het team staat niet open voor feedback of verbetersuggesties van buiten maar zoekt deze niet actief op.
  - **Versiebeheer**
    - ☐ Gepubliceerde versies zijn van elkaar te onderscheiden door consequente versienummering.
    - ☐ Als wijzigingen in de broncode of documentatie een opvolging zijn van een of meer gestelde vragen, wordt daar een referentie van opgenomen in het wijzigingsbericht. Er wordt daarmee erkenning gegeven aan degene die met de vraag of mogelijke verbeterpunten is gekomen.
  - **Documentatie**
    - ☐ Elke versie van de broncode bevat een Software Bill of Materials (SBoM) waarin in ieder geval de volgende informatie is opgenomen:
      - Naam van het component.
      - Waar de broncode van de component gevonden kan worden.
      - Licentie van het component.
      - Wie de IE-rechten houder is.
    - ☐ Er wordt in de README duidelijk toegelicht dat externe bijdragen niet in behandeling worden genomen.
    - * Er is goed gedocumenteerd hoe externe geïnteresseerden een bijdrage kunnen doen aan of vragen kunnen stellen over de broncode en/of documentatie.
    - ☐ In de root van de repository is een platte tekst bestand toegevoegd genaamd `CONTRIBUTING` (met of zonder txt of md extensie).
      - ☐ De conventies waaronder code-style, versie nummering, git workflow e.d. zijn duidelijk beschreven.
      - ☐ Het is duidelijk beschreven of en hoe er op de conventies zal worden gehandhaafd. Richt hier idealiter een geautomatiseerde zogenoemde linter voor in. Deze controleert automatisch op de navolging van de conventies.
      - ☐ Verwijs naar een technische referentie architectuur (TRA) wanneer deze beschikbaar is.

    - ☐ Maak duidelijk aan welke regels een verbeterverzoek moet voldoen. Denk daarbij aan geautomatiseerde conventie checks, een succesvolle CI/CD check, voldoen aan alle unittests, minimale code-coverage, reviews etc.
    - ☐ Er wordt uitgelegd hoe een bijdrager alle regels in de lokale context kan verifiëren.
  - ☐ In de root van de repository is een platte tekst bestand toegevoegd genaamd pull_request_template.md. De inhoud van dit bestand wordt getoond bij het openen van een merge/pull-request. Zo worden bijdragers bewust gemaakt van de projectregels bij het indienen van een verbetersuggestie.
- **Verbeterverzoeken**
  - ☐ Het proces van het verwerken van verbeterverzoeken is hetzelfde voor derden als voor teamleden.
  - ☐ Alle verbeterverzoeken worden onderworpen aan een code-review.
  - ☐ Bij het verwerpen van een verbeterverzoek wordt duidelijk waarom het verbeterverzoek is verworpen. Als het verbeterverzoek alleen geaccepteerd kan worden na wijzigingen, start dan een review proces zodat de verzoeker de kans krijgt zijn verzoek aan te passen.
  - ☐ Geef de indiener van een verbeterverzoek de mogelijkheid eerst akkoord te geven voor de daadwerkelijke opname.
  - ☐ Verbeterverzoeken worden pas geaccepteerd zodra aan alle voorwaarden is voldaan waaronder de test-coverage.

Ambitie 3. Actief samenwerken

- De broncode is in openbaarheid voor hergebruik gepubliceerd en het team staat open voor feedback of verbetersuggesties van buiten spant zich ervoor in om een community samenwerking op gang te brengen en te houden.
  - **Versiebeheer**
    - ☐ Als wijzigingen in de broncode of documentatie een opvolging zijn van een of meer gestelde vragen, wordt daar een referentie van opgenomen in het wijzigingsbericht. Er wordt daarmee erkenning gegeven aan degene die met de vraag of mogelijke verbeterpunten is gekomen.
  - **Documentatie**

- ☐ Elke versie van de broncode bevat een Software Bill of Materials (SBoM) waarin in ieder geval de volgende informatie is opgenomen:
  - Naam van het component.
  - Waar de broncode van de component gevonden kan worden.
  - Licentie van het component.
  - Wie de IE-rechten houder is.
- ☐ Er is goed gedocumenteerd hoe externe geïnteresseerden een bijdrage kunnen doen aan of vragen kunnen stellen over de broncode en/of documentatie.
- ☐ Er is een Code of Conduct aan de repository toegevoegd die de gedragsregels beschrijven binnen de community.
- ☐ In de root van de repository is een platte tekst bestand toegevoegd genaamd `CONTRIBUTING` (met of zonder txt of md extensie).
  - ☐ De conventies waaronder code-style, versie nummering, git workflow e.d. zijn duidelijk beschreven.
  - ☐ Het is duidelijk beschreven of en hoe er op de conventies zal worden gehandhaafd. Richt hier idealiter een geautomatiseerde zogenoemde linter voor in. Deze controleert automatisch op de navolging van de conventies.
  - ☐ Verwijs naar een technische referentie architectuur (TRA) wanneer deze beschikbaar is.
  - ☐ Maak duidelijk aan welke regels een verbeterverzoek moet voldoen. Denk daarbij aan geautomatiseerde conventie checks, een succesvolle CI/CD check, voldoen aan alle unittests, minimale code-coverage, reviews etc.
  - ☐ Er wordt uitgelegd hoe een bijdrager alle regels in de lokale context kan verifiëren.
- ☐ In de root van de repository is een platte tekst bestand toegevoegd genaamd `pull_request_template.md`. De inhoud van dit bestand wordt getoond bij het openen van een merge/pull-request. Zo worden bijdragers bewust gemaakt van de projectregels bij het indienen van een verbetersuggestie.

– **Verbeterverzoeken**

- ☐ Het proces van het verwerken van verbeterverzoeken is hetzelfde voor derden als voor teamleden.
- ☐ Alle verbeterverzoeken worden onderworpen aan een code-review.
- ☐ Maak bij het verwerpen van een verbeterverzoek duidelijk waarom het verbeterverzoek is verworpen. Als het verbeterverzoek alleen geaccepteerd kan worden na wijzigingen, start dan een review proces zodat de verzoeker de kans

krijgt zijn verzoek aan te passen.
- ☐ Geef de indiener van een verbeterverzoek de mogelijkheid eerst akkoord te geven voor de daadwerkelijke opname.
- ☐ Alle bijdragen vereisen een ondertekende CLA of een geaccordeerde DCO voordat ze worden geaccepteerd.
- ☐ Verbeterverzoeken worden pas geaccepteerd zodra aan alle voorwaarden is voldaan waaronder de test-coverage.

– **Community management**
- ☐ Er is een community-manager aanwezig als aanspreekpunt voor externen.
- ☐ Er wordt gebruik gemaakt van bestaande communities en communicatiekanalen.
- ☐ Een klankbordgroep met belangrijkste stakeholders is ingericht.
- ☐ De community wordt actief betrokken bij denkprocessen en achterliggende overwegingen.
- ☐ Er vinden periodieke bijeenkomsten plaats met de kerngroep, klankbordgroep en/of community.
- ☐ Het is vastgelegd hoe community-input wordt meegenomen in besluitvorming (bijv. roadmap).
- ☐ De interne en externe besluitvorming zijn zoveel mogelijk met elkaar verweven.

Veilig & Betrouwbaar

Ambitie 1 Basisveiligheid

- **Informatiebeveiliging**
  - ☐ De code voldoet aan de ICT-beveiligingsrichtlijnen voor web- en mobiele applicaties van het NCSC.
  - ☐ Extern gebruikte componenten worden periodiek bijgewerkt naar de laatste stabiele versie.
  - ☐ Er is een Coordinated Vulnerability Disclosure beleid aanwezig dat duidelijk toelicht hoe derden kwetsbaarheden kunnen melden.
- **Testen**
  - ☐ Elk verbeterverzoek wordt onderworpen aan unittests, geautomatiseerd via CI/CD.
  - ☐ Tests zijn reproduceerbaar door zowel interne als externe ontwikkelaars.
  - ☐ Van elke unittest wordt functioneel gedocumenteerd wat het doel is.

- ☐ Testresultaten worden zichtbaar gemaakt (bijv. met een badge).
- ☐ Er wordt gemonitord dat niet-publieke code niet onbedoeld openbaar wordt.

Ambitie 2 Proactieve veiligheid

- **Informatiebeveiliging**
  - ☐ De code voldoet aan de ICT-beveiligingsrichtlijnen voor web- en mobiele applicaties van het NCSC.
  - ☐ Extern gebruikte componenten worden periodiek bijgewerkt naar de laatste versie.
  - ☐ Er is een Coordinated Vulnerability Disclosure beleid aanwezig dat duidelijk toelicht hoe derden kwetsbaarheden kunnen melden.
  - ☐ Wijzigingen aan de code worden onderworpen aan een statische code-analyse (bugs, security, maintainability).
  - ☐ Er wordt in de broncode onderbouwd waarom bepaalde stukken code niet worden getest of door wie en per wanneer deze alsnog worden getest.
  - ☐ De test-coverage mag niet dalen tijdens de ontwikkeling van de broncode.
  - ☐ De broncode en alle afhankelijkheden worden periodiek gescanned op kwetsbaarheden en tenminste voor elke release.
  - ☐ Kwetsbaarheden worden afhankelijk van de ernst tijdig gepatched.
  - ☐ Branches die in productie of release gaan zijn beveiligd via branch protection rules zodat merges alleen mogelijk zijn wanneer:
    - ☐ Alle CI/CD tests geslaagd zijn.
    - ☐ Security scans (statische code-analyse, dependency scans) geslaagd zijn.
    - ☐ Code-reviews door aangewezen reviewers zijn afgerond.
  - ☐ Alle commits worden ondertekent met een sleutel uniek per ontwikkelaar.
- **Testen**
  - ☐ Elk verbeterverzoek wordt onderworpen aan unittests, geautomatiseerd via CI/CD.
  - ☐ Tests zijn reproduceerbaar door zowel interne als externe ontwikkelaars.
  - ☐ Van elke unittest wordt functioneel gedocumenteerd wat het doel is.
  - ☐ Testresultaten worden zichtbaar gemaakt (bijv. met een badge).
  - ☐ Er wordt gemonitord dat niet-publieke code niet onbedoeld openbaar wordt.
- **Verbeterverzoeken**

- ☐ Verbeterverzoeken worden pas geaccepteerd zodra aan alle voorwaarden is voldaan waaronder de test-coverage.

Ambitie 3 Volledig robuust

- **Informatiebeveiliging**

☐ De code voldoet aan de ICT-beveiligingsrichtlijnen voor web- en mobiele applicaties van het NCSC.

- ☐ Extern gebruikte componenten worden periodiek bijgewerkt naar de laatste versie.
- ☐ Er is een Coordinated Vulnerability Disclosure beleid aanwezig dat duidelijk toelicht hoe derden kwetsbaarheden kunnen melden.
- ☐ Wijzigingen aan de code worden onderworpen aan een statische code-analyse (bugs, security, maintainability).
- ☐ Er wordt in de broncode onderbouwd waarom bepaalde stukken code niet worden getest of door wie en per wanneer deze alsnog worden getest.
- ☐ De test-coverage mag niet dalen tijdens de ontwikkeling van de broncode.
- ☐ De broncode en alle afhankelijkheden worden periodiek dagelijks gescanned op kwetsbaarheden en tenminste voor elke release.
- ☐ Kwetsbaarheden worden afhankelijk van de ernst tijdig gepatched.
- ☐ Traceerbaarheid van kwetsbaarheden en patches:
  - ☐ Elke gevonden kwetsbaarheid wordt (automatisch) als issue in de repository geregistreerd.
  - ☐ Het issue bevat: datum van ontdekking, ernst, verantwoordelijk ontwikkelaar, en deadline voor patch.
  - ☐ Sluiten van het issue mag alleen na verificatie van de patch door CI/CD of code-review.
- ☐ Automatische alerts bij security regressies of daling van test-coverage.
- ☐ Branches die in productie of release gaan zijn beveiligd via branch protection rules zodat merges alleen mogelijk zijn wanneer:
  - ☐ Alle CI/CD tests geslaagd zijn.
  - ☐ Security scans (statische code-analyse, dependency scans) geslaagd zijn.
  - ☐ Code-reviews door aangewezen reviewers zijn afgerond.
- ☐ Alle commits worden ondertekent met een sleutel uniek per ontwikkelaar.

- **Testen**

- ☐ Elk verbeterverzoek wordt onderworpen aan unittests die geautomatiseerd gedraaid worden met gebruik van CI/CD straten.
- ☐ CI/CD-pijplijnen blokkeren merges als security- of coverage-streefwaarden niet gehaald worden.
- ☐ Tests zijn reproduceerbaar door zowel interne als externe ontwikkelaars.
- ☐ Van elke unittest wordt functioneel gedocumenteerd wat het doel is.
- ☐ Testresultaten worden zichtbaar gemaakt (bijv. met een badge).
- ☐ Er wordt gemonitord dat niet-publieke code niet onbedoeld openbaar wordt.

- **Verbeterverzoeken**
  - ☐ Verbeterverzoeken worden pas geaccepteerd zodra aan alle voorwaarden is voldaan waaronder de test-coverage.

Efficiënt & Onafhankelijk

Ambitie 1 De broncode wordt onder verantwoordelijkheid van één partij ontwikkeld.

- **Beheer en onderhoud**
  - ☐ Ondersteuning geleverd op basis van best-effort.
  - ☐ Er wordt op een git-ondersteunend platform gewerkt.
- **Issue management**
  - ☐ Derden kunnen in openbaarheid meldingen doen over de werking van de broncode.
- **Modulair werken**
  - ☐ Data en configuratie worden gescheiden gehouden van de rest van de broncode.

Ambitie 2 De broncode wordt in samenwerking met meerdere partijen ontwikkeld.

- **Beheer en onderhoud**
  - ☐ Ondersteuning geleverd op basis van best-effort.
  - ☐ Er staat duidelijk in de README beschreven hoe het beheer vormgegeven is.
  - ☐ Het beheer wordt onder gezamenlijke verantwoordelijkheid uitgevoerd.
  - ☐ Er is een gezamenlijke stuurgroep die toeziet het beheer.

- ☐ Er wordt op een git-ondersteunend platform gewerkt.

- **Issue management**
  - ☐ Derden kunnen in openbaarheid meldingen doen over de werking van de broncode.
- **Modulair werken**
  - ☐ Data en configuratie worden gescheiden gehouden van de rest van de broncode.
  - ☐ Er wordt met een software-architectuur inzichtelijk gemaakt hoe modules samen het totale systeem vormen.
  - ☐ Modules worden in samenwerking functioneel bedacht en door een of meerdere partijen uit de samenwerking ontwikkeld.
  - ☐ Modules zijn eenvoudig te her te gebruiken en te integreren in verschillende technische stacks. Denk daarbij aan micro-services of shared-libraries.

Ambitie 3 De broncode is ondergebracht bij een organisatie die toeziet op de gezamenlijke doorontwikkeling.

- **Beheer en onderhoud**
  - ☐ Er staat duidelijk in de README beschreven hoe het beheer vormgegeven is.
  - ☐ Het beheer wordt onder gezamenlijke verantwoordelijkheid uitgevoerd.
  - ☐ Er is een gezamenlijke stuurgroep die toeziet het beheer.
  - ☐ De stichting voert regie en ziet toe op het beheer.
  - ☐ Er wordt op een git-ondersteunend platform gewerkt, die wordt aangeboden of onder eigen beheer gehost bij, soevereine (publieke) organisaties
- **Issue management**
  - ☐ Er wordt aan derden de mogelijkheid geboden in openbaarheid meldingen te doen over de werking van de broncode.
  - ☐ Het beheer op de issues wordt door de onafhankelijke stichting gevoerd
- **Modulair werken**
  - ☐ Data en configuratie worden gescheiden gehouden van de rest van de broncode.
  - ☐ Er wordt met een software-architectuur inzichtelijk gemaakt hoe modules samen het totale systeem vormen.

- ☐ Modules worden in samenwerking functioneel bedacht en door een of meerdere partijen uit de samenwerking ontwikkeld.
- ☐ Modules zijn eenvoudig te her te gebruiken en te integreren in verschillende technische stacks. Denk daarbij aan micro-services of shared-libraries.

Aanbevelingen

In dit hoofdstuk zijn twee typen aanbevelingen te vinden: aanbevelingen die bedoeld zijn als suggestie en aanbevelingen die bedoeld zijn als waarschuwing.

- De **suggesties** bevatten tips die ter overweging meegenomen kunnen worden in de opdracht.
- De **waarschuwingen** bevatten overwegingen die de opdrachtgever bewust maken van belangrijke punten die zware consequenties kunnen hebben.

Het is belangrijk te realiseren dat de overheid sowieso een inspanningsverplichting heeft om proactief open source de broncode te publiceren waarvan zij de IE-rechten bezit. De aandachtspunten die deze plicht met zich meebrengt, worden ook in dit hoofdstuk besproken.

Deze aanbevelingen zijn van toepassing op het werken met open source bij nieuwe projecten, maar ook bij projecten die achteraf open source worden gepubliceerd, bijvoorbeeld als gevolg van een Woo-verzoek (passieve informatieverstrekking).

❗ Gedeeltelijke publicatie

Er zijn verschillende redenen waarom een (open source) softwareproject niet volledig (open source) gepubliceerd kan worden:

- Er is een absolute en/of relatieve uitzonderingsgrond van toepassing vanuit de Wet open overheid.
- Een open source publicatie kan worden gezien als een economische activiteit onder de Wet markt en overheid.
- Niet alle IE-rechten zijn in bezit van degene die wil publiceren.
- Delen van de broncode bevatten vertrouwelijke informatie.

Welke belemmering er ook is, publiceer altijd de onderdelen die wel open source gepubliceerd kunnen worden. Vertrouwelijke informatie kan zwartgelakt ('zwartlakken') worden, of de belemmering kan worden weggenomen.

Let op: een beroep op vertrouwelijkheid kan niet zomaar worden gedaan. Vertrouwelijkheid geldt nooit automatisch voor het hele project. Een claim op vertrouwelijkheid moet altijd gerelateerd zijn aan een van de uitzonderingsgronden in de Woo en goed worden gemotiveerd. Wees daarnaast transparant over welke uitzonderingsgronden zijn toegepast bij het niet volledig publiceren van de broncode.

❗ Absolute en relatieve uitzonderingsgrond

Soms hoeft de overheid informatie niet openbaar te maken. Voor bepaalde informatie bestaan uitzonderingsgronden. Denk bijvoorbeeld aan informatie die de veiligheid van de Staat in gevaar kan brengen. Deze uitzonderingsgronden zijn er in twee vormen:

Absolute uitzonderingsgronden (Woo, art. 5.1, eerste lid)

Bij deze gronden vindt geen belangenafweging plaats. Als overheidsinformatie een van de onderstaande gronden raakt, mag deze in geen geval worden gepubliceerd naar aanleiding van een Woo-verzoek:

- Eenheid van de Kroon
- Veiligheid van de Staat
- Vertrouwelijk meegedeelde bedrijfs- en fabricagegegevens
- Persoonsgegevens
- Nummers ter identificatie

Relatieve uitzonderingsgronden (Woo, art. 5.1 tweede lid)

Hier is wél een afweging van belangen nodig, bij voorkeur samen met juristen. Het publiceren van bepaalde informatie kan bijvoorbeeld marktverstorend werken. Voorbeelden van belangen die hierbij een rol kunnen spelen zijn:

- Bedrijfs- en fabricagegegevens
- Economische en financiële belangen van de overheid
- Goed functioneren van overheidsorganen

Als er uitzonderingsgronden van toepassing zijn, betekent dit niet dat er niets gepubliceerd mag worden. Alleen de onderdelen die onder de uitzondering vallen mogen zwartgelakt worden of

buiten de publicatie blijven. Zorg dat in de documentatie duidelijk staat welke delen zwartgelakt zijn of niet gepubliceerd, en waarom. Beschrijf ook de functie van die onderdelen, zodat derden de functionaliteit in de lokale context kunnen begrijpen en eventueel zelf invullen.

❗ Transparantie, ja, hergebruik, nee

Vanuit de Wet open overheid (Woo) geldt er een inspanningsverplichting om broncode proactief openbaar te maken. De Wet hergebruik overheidsinformatie (Who) schrijft daar bovenop voor dat openbaar gemaakte broncode ook zoveel mogelijk geschikt moet worden gepubliceerd voor hergebruik. De open source licentie wordt erin aangewezen als hét instrument om dat hergebruik te bereiken.

Er kunnen omstandigheden zijn waarin broncode wél openbaar wordt gemaakt voor transparantiedoeleinden, maar niet voor hergebruik. Dit kan zich voordoen wanneer openbaarmaking voor hergebruik de economische belangen van de overheid zou schaden. In dat geval is sprake van een relatieve uitzonderingsgrond als bedoeld in artikel 5.1, tweede lid, van de Woo.

In zulke situaties kan broncode worden gepubliceerd zonder open-sourcelicentie, met de vermelding dat alle rechten zijn voorbehouden. De code is dan wel inzichtelijk, maar mag niet worden hergebruikt. Het opnemen van dit voorbehoud is essentieel, omdat volgens artikel 15b Auteurswet openbaar gemaakte werken van de overheid, zonder expliciet voorbehoud, vrij gebruikt en gekopieerd mogen worden door derden. De overheid behoudt altijd het auteursrecht, maar zonder voorbehoud kan dit recht niet effectief worden gehandhaafd. Door “Alle rechten voorbehouden” te vermelden, wordt hergebruik zonder toestemming uitgesloten. Neem deze vermelding idealiter ook in alle broncodebestanden op, samen met een copyrightvermelding. Bijv:

```
/*
  Alle rechten voorbehouden
  Copyright Ministerie van Volksgezondheid, Welzijn en Sport
*/
```

Bij het via open source beschikbaar stellen van software moet o.a. gekeken worden naar de Wet Markt en Overheid (Wet M&O). De overheid kan door het beschikbaar stellen van software concurreren met reeds in de markt aanwezige software. Als dit het geval is, moet de overheid

zich houden aan de vier gedragsregels om concurrentievervalsing te voorkomen. Een van die gedragsregels schrijft voor dat de overheid alleen de broncode openbaar mag maken via open source als de integrale kosten worden doorberekend aan de eindgebruiker. Met de integrale kosten wordt bedoeld alle kosten die samenhangen met het aanbieden van de software aan de eindgebruiker. Denk daarbij aan bijvoorbeeld publicatie – en ontwikkelingskosten.

Of de overheid zich moet houden aan de (vier gedragsregels uit de) Wet M&O zal per geval onderbouwd moeten worden. Daarbij zal de Wet M&O sneller van toepassing zijn als de overheid de broncode als product op de markt exploiteert dan wanneer de overheid de broncode enkel openbaar maakt ter inzage. Mocht er sprake zijn van een wettelijke verplichting voor de overheid tot openbaarmaking van de broncode op grond van de Wet open overheid (Woo) of de Wet hergebruik van overheidsinformatie (Who) zal goed gekeken moeten worden naar de onderlinge verhouding tussen deze wetten enerzijds en de Wet M&O anderzijds.

In sommige gevallen kan software die door een overheidsorgaan in het kader van haar overheidstaak wordt gebruikt buiten de werking van de Wet M&O vallen. Artikel 25h, tweede lid, van de Mededingingswet, biedt namelijk een mogelijke uitzondering voor levering van goederen of diensten door bestuursorganen als deze zijn bestemd voor de uitvoering van een publiekrechtelijke taak. Of bepaalde software daadwerkelijk onder deze uitzonderingsgrond valt, en derhalve open source beschikbaar kan worden gesteld, dient ook weer per geval onderbouwd te worden en bij twijfel altijd te worden beoordeeld door een jurist.

Tot slot is het van belang om te benoemen dat er op dit moment een wetsvoorstel ligt om hoofdstuk 4b van de Mededingingswet (waarin de Wet M&O is opgenomen) aan te passen zodoende dat het aanbieden van open source software wordt uitgezonderd van de gedragsregel integrale kostendoorberekening. Achtergrond van deze wijziging is dat overheden gemakkelijker de broncode van open software kunnen publiceren waarmee maatschappelijke voordelen gerealiseerd kunnen worden. Omdat open source software per definitie gratis is kan de gedragsregel ‘integrale kostprijs doorberekening’ een belemmering vormen voor een bestuursorgaan om daadwerkelijk tot publicatie over te gaan (bron: Memorie van toelichting, Wijziging van de Mededingingswet in verband met aanpassing van de bepalingen over de markt en overheid, 35985-1). Op dit moment is het onduidelijk of en wanneer deze wetswijziging wordt doorgevoerd.

❗ Wie bezit de IE-rechten?

Vanuit de Auteurswet bezit de maker van creatief werk automatisch het auteursrecht, tenzij hierover expliciet andere afspraken zijn gemaakt. Broncode valt onder creatief werk, dus het is belangrijk om goed vast te leggen wie (auteursrechtelijk) rechthebbende is van de geleverde broncode. Alleen de rechthebbende kan namelijk beslissen hoe (zijn deel van) de broncode gepubliceerd mag worden. Bedenk dat naast het auteursrecht met name ook het merkenrecht en databankenrecht in het spel kunnen zijn. In dit stuk wordt geconcentreerd op het auteursrecht, aangezien veel open source licenties het merkenrecht ook standaard al meenemen in hun voorwaarden.

Veel grote softwareprojecten bestaan uit verschillende componenten. Elk component kan meerdere ontwikkelaars hebben, die elk een deel van de (auteurs)rechten bezitten. Het is daarom essentieel om inzicht te hebben in de (auteurs)rechten op de broncode of in ieder geval zekerheid te hebben dat alle onderdelen onder een open source licentie zijn vrijgegeven. Zo worden (auteursrecht)schendingen zoveel mogelijk voorkomen.

Hou in ieder geval rekening met de volgende aandachtspunten als je aan de slag gaat met opensourcewerken:

- **Externe medewerkers** Wanneer externen werken aan broncode, maak dan expliciet dat de volledige rechten (auteursrechten en eventuele merkenrechten, databankrechten en octrooirechten) bij de opdrachtgever komen te liggen. Bepalingen uit de standaard (inkoop)voorwaarden bieden daar onvoldoende juridische basis voor (bron).
- **Overdracht van broncode** Zorg dat je beschikt over de volledige rechten (auteursrechten en eventuele merkenrechten, databankrechten en octrooirechten) als je de broncode wil overdragen of wanneer je broncode overgedragen krijgt. Uitzondering is die broncode waar een open source licentie aan de broncode verbonden is. De overdracht van genoemde rechten moet altijd schriftelijk gebeuren daar waar van toepassing met een getekende overeenkomst en/of inschrijving in een relevant register. Als de rechten eerder niet zijn overgedragen, maar dat wel gewenst was, probeer deze alsnog te verkrijgen of een open source licentie aan de broncode te verbinden.

❗ Pseudonieme ontwikkelaars

Bij open source ontwikkelen in openbaarheid is het belangrijk goed na te denken over openbare bekendheid van ontwikkelaars.

Ook bij het achteraf publiceren is het belangrijk hierover afspraken te maken. Ontwikkelaars kunnen vanaf het begin onder een pseudoniem of nickname werken, of – als dat is vergeten – achteraf worden gepseudonimiseerd via de commit history. Let er bij achteraf pseudonimiseren op dat de commit history buiten de pseudonimiseringsslag intact blijft.

Gebruik pseudoniemen consequent binnen één project en zorg dat deze intern wel herleidbaar zijn tot een specifieke ontwikkelaar of (recht)spersoon. Mocht er rond een specifiek onderdeel van de broncode een geschil ontstaan, dan is het belangrijk te weten welke (rechts)persoon verantwoordelijk was voor dat onderdeel. Let op: per project betekent niet hetzelfde als per repository.

Ontwikkelen in openbaarheid biedt veel ontwikkelaars bovendien de kans om een openbaar portfolio op te bouwen. Ga er daarom niet automatisch van uit dat zij onder een pseudoniem willen werken; sommige ontwikkelaars vinden het juist waardevol om onder hun eigen naam bekend te staan.

⚠️ Gelijkblijvende of stijgende test-coverage

Het is niet altijd eenvoudig om bij de doorontwikkeling van broncode de test-coverage gelijk te houden of te verbeteren. Soms is snelheid belangrijker dan een sluitende test-coverage, bijvoorbeeld bij het snel beschikbaar krijgen van kritische bugfixes. In zulke gevallen kan de nieuwe code de algemene test-coverage tijdelijk onder de gewenste drempelwaarde brengen, wat niet wenselijk is.

Het is aan te raden om in deze situaties de metadata-mogelijkheden te gebruiken die in veel programmeertalen en test-coverage tools aanwezig zijn. Sluit de code die tijdelijk bewust niet getest wordt uit van de test-coverage, zodat het overall percentage niet daalt. In LCOV voor de C-programmeertaal kan dit bijvoorbeeld door code tussen `/*LCOV_EXCL_START*/` en `/*LCOV_EXCL_STOP*/` te plaatsen, maar dit verschilt per programmeertaal en tool.

Documenteer daarnaast wie wanneer verantwoordelijk is om de test-coverage alsnog op orde te brengen. Gebruik hiervoor gangbare tags zoals `TESTME`, `FIXME` of `TODO`. Zo weten derden dat de nieuwe code bewust niet getest is, en kan gecontroleerd worden of de ontbrekende tests tijdig worden toegevoegd.

💡 De ambitieladder als groeimodel

De ambitieladder kan worden gebruikt als groeimodel. Het is daarbij belangrijk bewust te zijn van onomkeerbare stappen die sommige ambitieniveaus van elkaar onderscheiden. Lagere niveaus kunnen soms stappen toestaan die vanuit hogere niveaus niet toegestaan zouden zijn. Dit speelt bijvoorbeeld bij Transparantie en Vertrouwen. Een belangrijk aandachtspunt is de integriteit van de historie: als deze eenmaal is gesquashed, is de integriteit niet meer gegarandeerd zoals bij een hoger ambitieniveau vereist.

Wil je de ambitieladder als groeimodel gebruiken, dan kan het in sommige gevallen beter zijn een hoger ambitieniveau iets af te zwakken. Bijvoorbeeld: zet volledig in op alle eisen van ambitieniveau 3 van Transparantie en Vertrouwen, maar spreek af dat vanaf het begin niet volledig in openbaarheid hoeft te worden ontwikkeld. Mocht later worden besloten alsnog volledig in openbaarheid te werken, dan kan dat vanaf dat moment worden opgepakt alsof er vanaf het begin op die manier is gewerkt.

Bij Veilig & Betrouwbaar speelt dit probleem van onomkeerbare stappen niet. Daardoor is het makkelijker om in de loop van de tijd door te groeien naar een hoger niveau.

💡 Actief samenwerken

Het ambitieniveau Actief samenwerken verschilt qua technische eisen niet veel van het lagere ambitieniveau Externe bijdragen worden behandeld maar niet actief gezocht. Het verschil zit vooral in activiteiten rondom de codebase, niet in de code zelf. Actieve samenwerking kan zelfs plaatsvinden zonder dat er nog een codebase bestaat, bijvoorbeeld tijdens de oriëntatie- of onderzoeksfase van een project.

Bij actieve samenwerking spelen vaak ook stakeholdersmanagement en communitybuilding een belangrijke rol. Daarnaast is een proactieve communicatiestrategie essentieel.

Een aantal acties die een organisatie kan ondernemen om actieve samenwerking te bevorderen:

- Bepaal wie de belangrijkste gebruikers, belanghebbenden of andere externe doelgroepen zijn en benader hen actief.
- Wijs iemand in het team aan voor contact met externen.
- Verweef interne en externe besluitvorming, zodat het ontwikkelteam externe input optimaal kan benutten.

- Zorg voor duidelijkheid, regelmaat en ritme, bijvoorbeeld door een kerngroep of klankbordgroep periodiek input te laten geven.
- Deel niet alleen eindresultaten, maar ook vraagstukken en deelresultaten.
- Houd rekening met de agenda en motivatie van externen; wees flexibel en positief, zeker bij vrijwillige bijdragen.
- Maak gebruik van bestaande communities en bestaande communicatiekanalen.
- Wees expliciet over projectdoelen, eigenaarschap, planning en gedragsregels.
- Laat zien hoe externe input invloed heeft op de uiteindelijke koers van het project.
- Beschrijf het onboarding proces voor nieuwe medewerkers.

❗ Wel of geen Contributors License Agreement (CLA) of Developer Certificate of Origin (DCO)?

Bij open source projecten komt het vaak voor dat meerdere partijen bijdragen aan dezelfde code. Het is dan handig om van tevoren duidelijk te maken hoe deze bijdrages juridisch worden geregeld. Dat hoeft niet ingewikkeld te zijn, maar helpt wel om misverstanden te voorkomen over wie wat mag doen met de code en onder welke voorwaarden.

Contributors License Agreement (CLA)

Een Contributors License Agreement (CLA) is een manier om afspraken vast te leggen tussen een bijdrager en het project. In de overeenkomst staat dat de bijdrager toestemming heeft om zijn bijdrage te doen, onder de voorwaarden die het ontvangende project stelt. Het intellectueel eigendom van de code blijft altijd bij de bijdrager.

Een CLA vangt over het algemeen twee zaken af: * Het geeft het project de vrijheid om later van licentie te wisselen zonder iedere bijdrager opnieuw om toestemming te hoeven vragen. Zonder CLA zou dat wél moeten, wat bij onbereikbare of weigerende bijdragers kan leiden tot een zogenoemde license lock-in. * Het biedt extra zekerheid tegen onverwachte auteursrechtschendingen. De bijdrager verklaart namelijk dat hij gerechtigd is de bijdrage te leveren en daar de verantwoordelijkheid voor neemt.

Belangrijk: ook zonder CLA blijft de gekozen open source licentie altijd leidend. De CLA is dus een aanvulling, geen vervanging.

Developer Certificate of Origin (DCO)

Een andere, lichtere manier om de juridische afspraken vast te leggen is het Developer Certificate of Origin (DCO). Bij een DCO verklaart een bijdrager bij elke bijdrage dat hij het

recht heeft om de code bij te dragen en dat hij akkoord gaat met de licentie van het project. Dit gebeurt vaak via een eenvoudige check bij het indienen van een commit of pull-request. Het voordeel is dat het minder formeel is dan een CLA - hij hoeft immers niet ondertekent te worden - maar toch voldoende zekerheid biedt dat de bijdragen correct zijn afgegeven.

Kiezen tussen CLA en DCO

Projecten kunnen kiezen voor een CLA, een DCO, beide of geen van beide, afhankelijk van de omvang, complexiteit en juridische gevoeligheid van het project. Wat belangrijk is, is dat de regels duidelijk zijn voor iedereen die bijdraagt. Zo weet een externe bijdrager precies onder welke voorwaarden zijn code wordt geaccepteerd en voorkomt het dat later discussies ontstaan over rechten of gebruik.

De Auteursketen in de EUPL

De Europese Unie Public Licence (EUPL) v1.2 bevat in artikel 6 een bepaling die vaak de auteursketen wordt genoemd. Daarin staat dat iedere licentiegever – dus ook iedere bijdrager – garandeert dat hij de rechthebbende is op de code of bevoegd is deze te licentiëren. Met andere woorden: wie bijdraagt onder de EUPL verklaart automatisch dat hij het recht heeft om zijn bijdrage beschikbaar te stellen en dat deze onder de EUPL mag worden verspreid.

Hiermee voorziet de EUPL in dezelfde zekerheid die een DCO biedt. Voor projecten die onder de EUPL werken, is een aparte DCO dus in beginsel niet nodig. Alleen wanneer een project meer of andere rechten wil regelen – zoals de mogelijkheid om van licentie te wisselen of aanvullende vrijwaringen – kan een aparte CLA nog zinvol zijn.

Geen CLA of DCO

Zelfs zonder CLA of DCO is het dus goed mogelijk om een open source project te beheren. In dat geval geldt automatisch dat bijdragers akkoord gaan met de open source licentie van het project, maar het kan handig zijn dit expliciet te vermelden in de documentatie of in een pull request template, zodat het voor iedereen duidelijk is.

Licenties en eigendom

Het is belangrijk te beseffen dat de projectlicentie niet automatisch geldt voor alle afzonderlijke componenten. Zeker niet wanneer je die componenten ook los van het project wilt hergebruiken. De projectlicentie bepaalt onder welke voorwaarden het geheel – de verzameling van alle onderdelen – wordt gepubliceerd. Zo kan een project in zijn geheel onder de EUPL staan, terwijl sommige losse bestanden of modules nog onder MIT of Apache hergebruikt kunnen worden.

Voor de delen van de code waar je zelf de IE-rechten van bezit, kun je altijd nog van licentie wisselen. Zodra de code echter is vermengd met bijdragen van derden, zit je vast aan de voorwaarden waaronder zij hun bijdrage hebben geleverd. Een licentiewijziging kan dan alleen met hun toestemming. Let er ook op dat een licentiewijziging van een module gevolgen kan hebben voor de licentie van het project als geheel. Zo kan een project als geheel niet langer onder de EUPL blijven, wanneer een losse module van MIT naar GPL wijzigt – althans niet meer in combinatie met die specifieke versie van de module.

Rationale

Op het moment van schrijven van dit document is het niet mogelijk om commentaar uit HackMD te exporteren. Het is dus niet makkelijk inzichtelijk te maken welke interacties (met wie) tot een bepaalde wijziging hebben geleid. Om die informatie niet verloren te laten gaan zal een samenvatting van die gesprekken, en de eventuele wijzigingen waartoe ze geleid hebben, hier weergegeven worden.

Bestandsformaten voor documentatie

10 augustus 2025

Net als software worden documenten opgemaakt volgens bepaalde afspraken. Zo is bij een tekstdocument aan de hand van metadata te achterhalen welk lettertype, welke lettergrootte, kleur en uitlijning gebruikt moet worden. Wanneer deze afspraken volledig openbaar en zonder belemmeringen zijn gedocumenteerd, spreken we van een open bestandsformaat. Voor tekstdocumenten zijn voorbeelden LaTeX, Typst het Open Document Format (ODF) ontwikkeld door OASIS.

Software wordt geschreven in platte-tekstbestanden in een programmeertaal; dit noemen we broncode. Het voordeel van platte tekst is dat het geopend kan worden zonder aanvullende programmatuur en er gemakkelijk in samen te werken is. In combinatie met versiebeheersystemen zijn verschillen tussen versies eenvoudig te zien. Voor tekstdocumenten bestaan bestandsformaten die deze voordelen naar documentatie brengen, zoals Markdown (MD; waarin deze ambitieladder is geschreven), maar ook RestructuredText (RST).

Deze ambitieladder adviseert om alle documentatie zoveel mogelijk in deze platte-tekstformaten op te stellen, met voorkeur voor Markdown, omdat dit de best ondersteunde standaard

is. Als platte-tekstdocumentatie niet mogelijk is, kies dan een open standaard die zonder belemmeringen beschikbaar is. Vermijd formaten zoals Open Office XML (OOXML) van Microsoft, dat vooral goed te openen is met betaalde software. Gebruik in plaats daarvan het Open Document Format (ODF; ODT, ODS), dat zowel door Microsoft Office als door LibreOffice of OpenOffice geopend kan worden.

Alleen als het echt niet mogelijk is om aan bovenstaande eisen te voldoen, mag documentatie in een ander formaat worden gedeeld. In dat geval moeten altijd de bronbestanden beschikbaar zijn. Denk bijvoorbeeld aan een architectuurplaat die is gemaakt in gesloten software en opgeslagen als afbeelding (JPG, PNG). Zo is de inhoud zonder belemmeringen te bekijken. Om de architectuurplaat aan te kunnen passen is vervolgens wel het bronbestand nodig waarin het is opgeslagen door de oorspronkelijke programmatuur. Ook deze bronbestanden moeten om die reden ook gedeeld worden.

Test-coverage

10 augustus 2025

Broncode bestaat uit geprogrammeerde logica, die samen de software vormt. Programmeren is grotendeels mensenwerk, en om zekerheid te hebben dat de logica correct is vertaald naar code, moet deze getest worden. Een simpel voorbeeld: bij een rekenmachine test je of de uitkomst van 1+1 daadwerkelijk 2 is. Bij grote softwareoplossingen is handmatig testen echter niet haalbaar, omdat honderden tests nodig zijn om alle delen van de code te valideren. Het testen van kleine stukjes logica wordt **unittests** genoemd; een unit is Engels voor een klein afgebakend blokje code.

Met **test-coverage** wordt aangegeven hoe omvattend de unittests zijn. Dit bestaat uit twee onderdelen:

- Of alle regels van een broncodebestand worden geraakt door een unittest.
- Of alle functies worden geraakt door een unittest.

Het is makkelijker om alle functies te testen dan alle regels. Een test-coverage van 100% betekent dat elke uithoek van de code wordt getest, maar dit garandeert niet altijd dat de tests functioneel zinnig zijn. Bijvoorbeeld: het tegelijkertijd indrukken van alle knoppen van een formulier is geen zinvolle test, omdat een normale gebruiker dat nooit zou doen. De daadwerkelijke werking voor de eindgebruiker wordt dan niet gevalideerd.

Test-coverage wordt doorgaans gezien als een kwaliteitskeurmerk van de broncode. Idealiter stijgt de test-coverage met de doorontwikkeling van de code, of daalt deze tenminste niet. In sommige sectoren zijn bepaalde tests verplicht:

- **Automotive**: de regel 'fail-safe' betekent dat een auto, ook bij een storing, veilig aan de kant gezet kan worden.
- **Luchtvaart**: de regel 'fail-operational' vereist dat bij falen van een onderdeel, een ander onderdeel de functie kan overnemen, zodat het vliegtuig blijft vliegen.

Test-coverage biedt meerdere voordelen:

- Je controleert of de aannames die tijdens het programmeren zijn gemaakt, kloppen.
- Je maakt deze aannames inzichtelijk voor derden.
- Je kunt bij verdere ontwikkeling controleren of nieuwe wijzigingen niet per ongeluk bestaande functionaliteit breken.

Voor opensourcewerken zijn vooral het tweede en derde voordeel belangrijk. Tests maken het voor derden makkelijker om bij te dragen, omdat de werking van de verbeteringen in de lokale context getest kan worden. Zo heb je meer zekerheid dat je verbetering geen onbedoelde schade aanricht. Als de verbeteringen zelf tests bevatten, kan de ontvangende partij gemakkelijk de intentie en correctheid van de wijziging controleren.

Hoe belangrijk tests ook zijn, er zijn situaties waarin de noodzaak voor nieuwe functionaliteit te groot is om uitgebreid stil te staan bij een sluitende test-coverage. Denk bijvoorbeeld aan het zo snel mogelijk repareren van een kritieke fout in de broncode. In dergelijke gevallen kan tijdelijk worden afgeweken van strikte drempelwaarden. Het blijft echter aan te raden om deze kritieke verbeteringen zo snel mogelijk alsnog te voorzien van unittests en andere relevante tests.

De integriteit van de history

10 augustus 2025

Om effectief opensourcewerken toe te passen, is het gebruik van gedistribueerde versiebeheersystemen cruciaal. Hierbij bestaat er één hoofdversie van de broncode, waarvan iedereen lokaal een kopie kan maken. In GitHub-termen staat deze versie meestal in de main branch. Lokale kopieën kunnen zich bevinden bij de ontwikkelaars van het project, maar

ook bij externen die willen bijdragen. Hierdoor kan iedereen een eigen versie van de broncode hebben, afhankelijk van de wijzigingen in zijn lokale kopie.

Tussentijds moet de lokale kopie natuurlijk worden gesynchroniseerd met de hoofdbroncode, zodat wijzigingen van anderen correct worden meegenomen. Voor een soepele synchronisatie is het essentieel dat de integriteit van de hoofdbroncode behouden blijft. Met andere woorden: als je gisteren alle wijzigingen hebt binnengehaald, mogen deze niet later door iemand anders worden overschreven of verstoord.

Een nuance vormt de ontwikkeling die buiten de hoofdbroncode plaatsvindt, bijvoorbeeld in onafhankelijke verbeterverzoeken of mini-projecten. In zulke gevallen is het minder belangrijk dat de historie volledig integer is. Soms is het zelfs wenselijk om de historie van een verbeterverzoek te herschikken om overzicht te behouden in de hoofdbroncode.

Iedereen kan asynchroon verbeteringen voorstellen. Deze voorstellen kunnen op zichzelf een rijke historie hebben. Wanneer al deze histories direct in de hoofdbroncode worden samengevoegd, kan het moeilijk zijn om te zien welk punt bij welke verbetersuggestie hoort. Het is dan vaak beter om alle wijzigingen samen te nemen en als één toevoeging in de hoofdbroncode op te nemen.

Ambities Veilig & Betrouwbaar

12 september 2025

In de oorspronkelijke opzet werd er geworsteld met een juiste verwoording van ambities voor Veilig & Betrouwbaar en het groeipad tussen de ambitieniveaus. De oorspronkelijke ambities waren deze:

> 1. De broncode bevat geen bedrijfskritische onderdelen, verwerkt geen gevoelige gegevens en maakt geen onderdeel uit van overheidscommunicatie.
> 2. De broncode is enigzins bedrijfskritisch, verwerkt enigzins gevoelige gegevens of wordt gebruikt in overheidscommunicatie.
> 3. De broncode is van grote bedrijfskritische waarde, verwerkt hoog gevoelige gegevens of wordt gebruikt voor kritische overheidscommunicatie.

Ook waren in de oorspronkelijke versie de verschillen tussen de ambities marginaal. Zo was het enige verschil tussen ambitieniveau 2 en 3 dat deze eis veranderde van een periodiek naar een dagelijks:

> De broncode en alle afhankelijkheden worden periodiek dagelijks gescanned op kwetsbaarheden en tenminste voor elke release.

Ook met de initiële opstellers van dit document kwamen we hier niet goed uit. We doorvoelde wel dat de verwoording anders was dan bij de andere categorieën en dat er niet een logische groei in zat. Door de hoofdauteur is dit dilemma vervolgens voorgelegd aan GenAI.

> Waar ik tegen aanloop is dat Veilig & Betrouwbaar niet echt verwoord is als ambities i.t.t. andere doelen. Heb je daar nog tips voor?

Het antwoord was in de kern:

> Ah, dat is een goed punt. Je ziet bij "Veilig & Betrouwbaar" dat de ambitieniveaus nu vooral beschrijvend zijn over het soort broncode ("niet kritisch", "enigszins kritisch", "kritisch"), terwijl bij de andere pijlers de ambitie meer gedrags- of procesmatig wordt geformuleerd (bijv. "Actief samenwerken" of "Periodiek publiceren").

Oftewel, een mooie verwoording van het dilemma. Het tweede probleem is ook aan GenAI voorgelegd:

> Het probleem is ook dat ambities maar beperkt verschillen.

Dit bleek wat lastiger te zijn, omdat GenAI sterk blijft hangen in het denkraam dat door de vraagsteller wordt aangedragen.

> Ah, precies! Dat is een klassiek issue bij beveiliging: er is een basisniveau dat sowieso al vereist is, en de verschillen tussen "matig kritisch" en "zeer kritisch" zijn vaak meer gradueel dan fundamenteel. Daardoor lijkt het alsof de ambitieniveaus nauwelijks verschillen, en dat kan het visuele en strategische onderscheid minder duidelijk maken.

Een van de suggesties van GenAI was het volgende:

> Focus op procesmatige volwassenheid Beveiliging kan moeilijk fundamenteel verschillen op code-niveau, maar wel op procesniveau. Voorbeelden:
>
> - Ambitie 1: Testen en patchen gebeurt reactief.
> - Ambitie 2: Testen en patchen gebeurt systematisch en gedocumenteerd.
> - Ambitie 3: Testen, patchen en compliance wordt continu gemonitord en geaudit.

De wedervraag:

> Is deze suggestie meer te formuleren als ambities zoals ook de rest van de categorieën?

Het antwoord:

> Ja, dat kan zeker!
>
> Het idee is: in plaats van te zeggen “de broncode is enigszins kritisch”, formuleer je wat de organisatie wil bereiken op veiligheidsniveau.
>
> Ambitie 1 – Basisveiligheid Ambitie 2 – Proactieve veiligheid Ambitie 3 – Volledig robuust

Zoals te zien is deze suggestie overgenomen. Vervolgens is samen met GenAI gekeken hoe de huidige eisen in dit groeimodel te passen zijn.

Een ander dilemma was dat in de initiële versie operationele streefwaarden werden weer opgevoerd:

> ☐ Wijzigingen aan de code worden onderworpen aan een statische code-analyse, met de volgende streefwaarden:
>
> ☐ Reliability 0 Bugs
> ☐ Maintainability < 5% debt ratio
> ☐ Security 0 Vulnerabilities
> ☐ Test-coverage > 80%

Deze streefwaarden kunnen alleen erg verschillen tussen organisaties. Het heeft dus niet zoveel zin om hier specifieke waarden te noemen. Het is vooral belangrijk dat er bewust wordt nagedacht over streefwaarden. Deze zijn in een latere versie dan ook weggelaten.

Met de nieuwe categorisering en de meer functioneel verwoording is uiteindelijk een meer logisch groeimodel tussen de verschillende ambities geformuleerd.

Bijdragers

Dank aan alle hier bij naam genoemd, maar ook alle bijdragers die graag anoniem willen blijven.

- Maurice Hendriks (Hoofdauteur; Ministerie van Volksgezondheid, Welzijn en Sport)
- Antoine den Brok (Curavista)
- Anne Jan Brouwer (Ministerie van Volksgezondheid, Welzijn en Sport)

- Mitch Hak (Ministerie van Volksgezondheid, Welzijn en Sport)
- Paul Hoogland (Stichting MedMij)
- Walter van Holst (Hooghiemstra & Partners)
- Ivo Jansch (Dawn Technology/HCI)
- Mark Janssen (Ministerie van Volksgezondheid, Welzijn en Sport)
- Bert van Kammen (Quli/HCI)
- Dirkjan Ochtman (XavaMedia)
- Davey Schilling (Topicus)
- Eva van Sloten (Ministerie van Volksgezondheid, Welzijn en Sport)
- Jan Verhoeckx (Topicus)
