Realisatie van QC-Wizard v2.0

Kwaliteitsborging grondwaterstands- en stijghoogtegegevens

1 Inleiding

1.1 Aanleiding en achtergrond

Doel van dit document is het ondersteunen van het platform meetnetbeheerders bij hun keuze van de verdere aanpak en uitwerking die nodig zijn om het QC-protocol te implementeren in software. Onder andere ten behoeve van de KRW rapportages dient de kwaliteit van grondwaterstands- en stijghoogtegegevens op orde te zijn. Eind september 2017 is QC-protocol versie 2.0 voor grondwaterstanden en stijghoogten vastgesteld in het platform meetnetbeheerders. Hiermee is de eerste stap gezet om de beoogde borging van de kwaliteit van onze meetgegevens ook in praktijk te brengen.

De uiteindelijke software dient op een goede manier ingebed te zijn in de werkprocessen van de provincies (en anderen), en de andere software die daarbij gebruikt wordt. Uitgangsmateriaal voor deze implementatie stap is het 'papieren' QC-protocol v2.0 en de QC-software die reeds ontwikkeld en beschikbaar is (Von Asmuth, 2018). Omdat de opzet en ervaringen met deze QC-Wizard v0.5 positief waren, wordt hier nader bekeken hoe deze omgevormd kan worden naar software die de meetnetbeheerder ondersteunt bij toepassing van QC protocol v2.0 in de praktijk, en de correcte toepassing borgt. Deze omvorming kan het beste gefaseerd opgepakt en gerealiseerd worden.

1.2 QC-Wizard v0.5: voorbeeld en evaluatiesoftware

QC-Wizard v0.5, die reeds ontwikkeld en beschikbaar is, is een nevenopbrengst van een pilotproject, waarin werkversie 1.0 van het protocol geëvalueerd is, en de softwarematige implementatie van het protocol is getest. De ervaringen die opgedaan zijn tijdens deze pilot hebben bovendien geleid tot een update van QC-Protocol versie 1.0, en uitgifte van versie 2.0. Doel van QC-Wizard v0.5 was het uitproberen en evalueren van verschillende controles en de implementatie van het QC-protocol in Wizard-vorm, tijdens een hands-on workshop met het platform meetnetbeheerders. QC-Wizard v0.5 is in de huidige vorm echter niet geschikt voor operationeel gebruik.

1.3 Gefaseerd van prototype naar definitief

De t.b.v. de pilot ontwikkelde QC-Wizard dient dus geschikt gemaakt te worden voor operationeel gebruik. Realisatie van deze doelstelling, brengt echter zowel inhoudelijke en functionele aspecten met zich mee, als aspecten op het ICT-gebied. Bij deze laatste gaat het bijvoorbeeld om de ICT-architectuur, het ontwikkelplatform en de communicatie met en inbedding in de andere, relevante programmatuur die gebruikt wordt bij grondwatermonitoring. Zolang de inhoudelijke en functionele kant van de QC-Wizard nog onvoldoende uitontwikkeld zijn, kan overschakeling op een andere ICT-architectuur dat proces compliceren en vertragen. Daarnaast speelt mee dat de BRO nog in ontwikkeling is, waar het definitieve datamodel en eventueel ook de ICT-architectuur van zal afhangen en op zal moeten wachten. In dit document wordt bij de verdere aanpak uitgegaan van de volgende fasering:

1) Realisatie van een werkend prototype van de operationele QC-Wizard v2.0, o.b.v. de huidige QC-Wizard v0.5

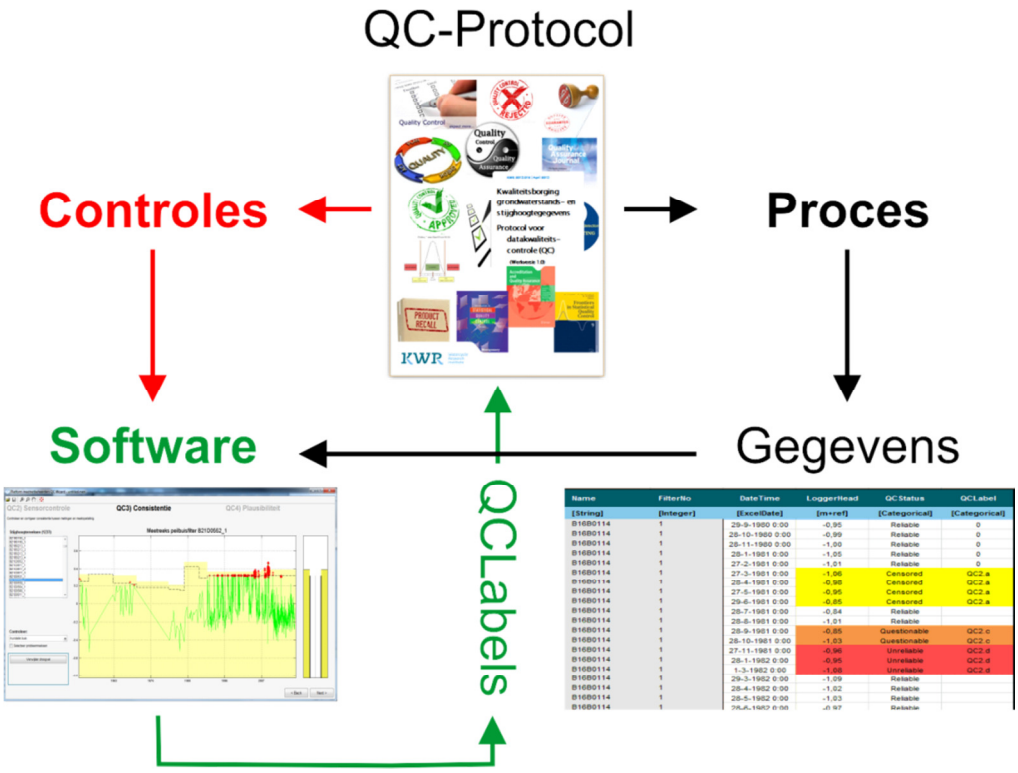
2) Verkenning van de definitieve ICT-architectuur, inbedding, beheer en onderhoud

3) Migratie van het prototype van QC-Wizard v2.0 naar de definitieve ICT-architectuur en beheer

2 Fase 1: QC-Wizard v2.x (prototype)

2.1 Werkproces, controles en software

Fase 1 van de voorgestelde aanpak is gericht op het realiseren van software (als prototype), die een gebruiker in staat stelt om de in protocol v2.0 beschreven controles uit te voeren wanneer het daar beschreven werkproces wordt gevolgd. Ook in het protocol zelf staat deze driedeling in werkproces, controles en software al centraal. Proces, controles en software zijn daarbij onlosmakelijk met elkaar verbonden (Figuur 1). Voor het uitvoeren van de controles is het noodzakelijk dat de benodigde gegevens voorhanden zijn, en op de voorgeschreven wijze verzameld. Om toepassing van de controles te faciliteren en uniformeren, dienen deze geïmplementeerd te zijn in een bijbehorende stuk software. Met behulp van deze software kunnen de gegevens gecontroleerd, gecorrigeerd en gekeurd worden conform de richtlijnen van het QC-protocol. Als resultaat daarvan worden de gegevens vervolgens aangerijkt met QC-labels, zoals beschreven in (Von Asmuth, 2018).



Figuur 1: Samenhang tussen controles en software, proces en gegevens, labels en protocol (Von Asmuth, 2018)

2.2 Functionele doelen en randvoorwaarden

De doelen en randvoorwaarden die belangrijk zijn bij de realisatie van een werkend prototype van de operationele QC-Wizard v2.0 zijn:

- QC-Wizard v2.0 dient geschikt gemaakt en ingericht te worden voor de eigenlijke werkprocessen van de provincies. Toepassing van het QC-protocol en de QC-Wizard moet leiden tot gegevens met een hogere en meer uniforme kwaliteit in minder tijd.

- QC-Wizard v2.0 dient QC-protocol v2.0 te volgen. De controles, richtlijnen en criteria van het protocol dienen daartoe één-op-één geïmplementeerd te worden in de software. De punten waar dit eventueel niet of niet eenvoudig mogelijk blijkt te zijn, dienen geïnventariseerd, geëvalueerd en gecommuniceerd te worden.

- De controle-algoritmen die in de QC-Wizard geïmplementeerd worden, dienen scherp en eenduidig te zijn. De controle-algoritmen dienen daartoe zoveel mogelijk de verschillende QC-Labels af te dekken. De in de gegevens voorkomende problemen of fouten dienen zoveel mogelijk automatisch gedetecteerd te worden. Andersom gezien dienen de controles zo min mogelijk gegevens onterecht als foutief te labellen.

- Om te zorgen dat andere softwareleveranciers (van de door provincies gebruikte databases en/of databeheersoftware) relatief eenvoudig en robuust kunnen aansluiten op en interfacen met de QC-Wizard, dient deze zoveel mogelijk gebruik te maken van open standaarden. De QC-Wizard dient, in het bijzonder, afgestemd te zijn op en te kunnen communiceren met de (nog in ontwikkeling zijnde) BRO

- Om de toepassing van het QC-protocol en het gebruik en de verdere ontwikkeling van de QC-Wizard te stimuleren dient deze vrij toegankelijk te zijn voor derden. Om hergebruik, leveranciersonafhankelijkheid en samenwerken aan de code van de QC-Wizard mogelijk te maken, en aldus efficiëntie en continuïteit te waarborgen, dient de code de QC-Wizard *open source* beschikbaar gemaakt te worden

2.3 Ontwikkelplatform en architectuur

Qua ontwikkelplatform en architectuur kan QC-Wizard v2.0 in eerste instantie de huidige QC-Wizard v0.5 volgen. Het gebruikte ontwikkelplatform daarvoor is MATLAB (zie bijlage A), het resultaat een desktop applicatie die lokaal geïnstalleerd en gebruikt kan worden. Door gebruik te maken van de reeds beschikbare software en code, zijn er in deze fase geen extra tijd of kosten gemoeid met de onderliggende ICT-aspecten. De tijd en aandacht kunnen in dat geval volledig uitgaan naar de benodigde uitwerking van de controles en functionaliteit van de Wizard. In een volgende fase zal de QC-Wizard v2.0 vervolgens relatief eenvoudig en naar wens gemigreerd kunnen worden naar de definitieve ICT-architectuur, door (een) nader te selecteren software-ontwikkelaar(s).

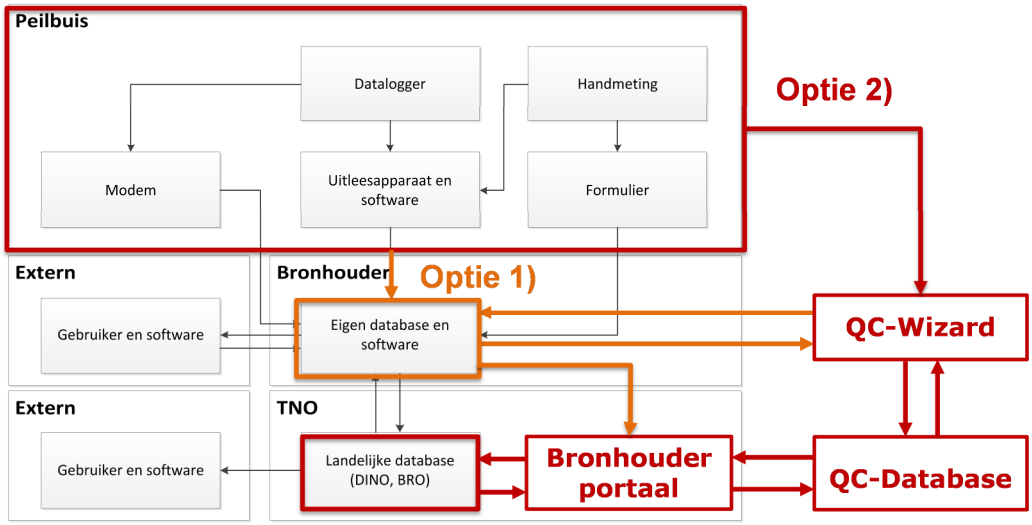
3 Fase 2/3: QC-Wizard v2.0 (definitieve versie)

3.1 Inbedding in de data- en workflow

Fase 2 en 3 van de voorgestelde aanpak zijn gericht op het inbedden van de QC-Wizard in de daadwerkelijke datastroom van de meetnetbeheerder, en de verschillende andere software en systemen die daarbij gebruikt worden. De datastroom van 'bron tot BRO' is in het kader van de kwaliteitsborgingsprojecten eerder beknopt beschreven en grafisch weergegeven in (Von Asmuth en Van Geer, 2015). We gebruiken de figuur uit dat rapport hier, om de beoogde rol van de QC-Wizard in de datastroom van bron tot BRO te verduidelijken (Figuur 2). Op hoofdlijnen zijn er daarbij twee opties, afhankelijk van de vraag of de datastroom van de meetnetbeheerder verloopt via:

1. Een eigen database en software
2. De QC-Wizard

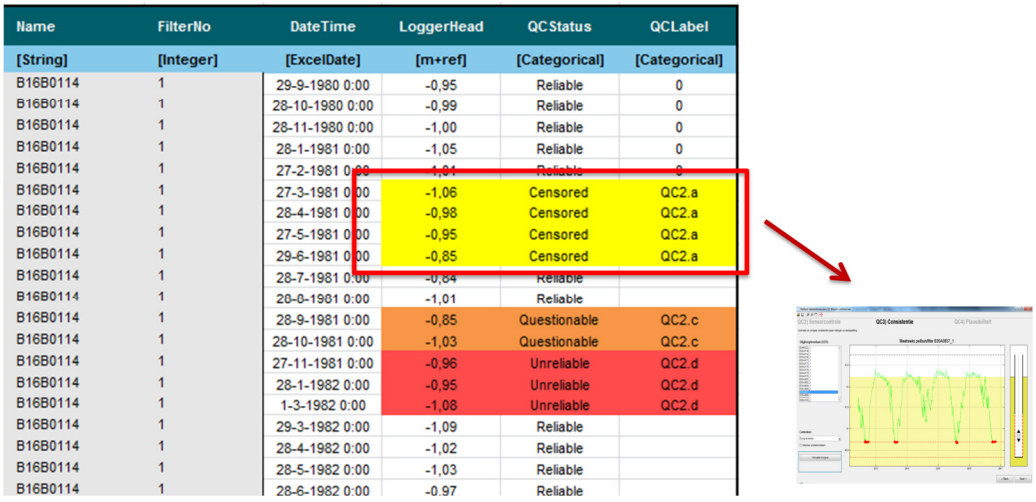
Eindresultaat van de QC-Wizard zijn de eventuele correcties, QC-Labels en QC-Status die aan de gegevens zijn toegekend (zie onderstaande tabel en/of QC-Protocol v2.0). De resultaten dienen uiteraard opgeslagen en uitgewisseld te kunnen worden. Een voorbeeld van uitwisseling van de resultaten is te vinden in Figuur 3. Opslag kan optioneel in de eigen database, of in een aparte en bij de QC-Wizard behorende QC-Database.



Figuur 2: Rol van de QC-Wizard in de datastroom van bron tot BRO (naar Von Asmuth en Van Geer, 2015)

Veld- of kolomnaam	Datatype	Omschrijving
QC-Label	Categorical	Controleresultaat en oorzaak van een kwaliteitsoordeel. Voor een overzicht en de betekenis van individuele labels, zie hoofdstuk 2.
QC-Status	Categorical	Status of eindoordeel over de kwaliteit van een meting, met als opties: - reliable (betrouwbaar) - questionable (twijfelachtig) - unreliable (onbetrouwbaar) - censored (gecensureerd) - estimated (geschat) - missing (ontbrekend)

Tabel 1: Toelichting op de tot QC-Protocol v2.0 behorende QC-Labels en QC-Status



Name	FilterNo	DateTime	LoggerHead	QCStatus	QCLabel
[String]	[Integer]	[ExcelDate]	[m+ref]	[Categorical]	[Categorical]
B16B0114	1	29-9-1980 0:00	-0,95	Reliable	0
B16B0114	1	28-10-1980 0:00	-0,99	Reliable	0
B16B0114	1	28-11-1980 0:00	-1,00	Reliable	0
B16B0114	1	28-1-1981 0:00	-1,05	Reliable	0
B16B0114	1	27-2-1981 0:00	-1,04	Reliable	0
B16B0114	1	27-3-1981 0:00	-1,06	Censored	QC2.a
B16B0114	1	28-4-1981 0:00	-0,98	Censored	QC2.a
B16B0114	1	27-5-1981 0:00	-0,95	Censored	QC2.a
B16B0114	1	29-6-1981 0:00	-0,85	Censored	QC2.a
B16B0114	1	28-7-1981 0:00	-0,84	Reliable	
B16B0114	1	28-8-1981 0:00	-1,01	Reliable	
B16B0114	1	28-9-1981 0:00	-0,85	Questionable	QC2.c
B16B0114	1	28-10-1981 0:00	-1,03	Questionable	QC2.c
B16B0114	1	27-11-1981 0:00	-0,96	Unreliable	QC2.d
B16B0114	1	28-1-1982 0:00	-0,95	Unreliable	QC2.d
B16B0114	1	1-3-1982 0:00	-1,08	Unreliable	QC2.d
B16B0114	1	29-3-1982 0:00	-1,09	Reliable	
B16B0114	1	28-4-1982 0:00	-1,02	Reliable	
B16B0114	1	28-5-1982 0:00	-1,03	Reliable	
B16B0114	1	28-6-1982 0:00	-0,97	Reliable	

Figuur 3: Voorbeeld van uitwisseling van gegevens die aangerijkt zijn met een QC-Status en QC-Label, via het HydroMonitor format (von Asmuth en Vonk, 2017).

3.2 Ontwikkelplatform, architectuur en onderhoud

Nadeel van een desktop applicatie, zoals QC-Wizard v0.5, is dat deze lokaal geïnstalleerd dient te worden en (in principe) alleen lokaal toegankelijk is. De QC-Wizard kan in de volgende fasen naar een online applicatie gemigreerd worden, al dan niet met behulp van een ander ontwikkelplatform. Hiermee wordt de QC-Wizard centraal beschikbaar gemaakt, voor iedereen die desgewenst toegang verleend wordt. Deze keuze heeft echter ook nadelen, bijvoorbeeld omdat de bijbehorende rechten- en veiligheidsaspecten daarmee ook nader bekeken en opgelost moeten worden. Een tussenstap of tussenvorm is ook mogelijk, bijv. in de vorm van een online QC-Database en een lokale QC-Wizard die dient als lokaal instrument en gebruikersinterface voor de meetnetbeheerder.

4 Sturing en risicomanagement

4.1 Fase 1: Stappen en tussenproducten

Een aanpak die binnen de ICT-wereld gangbaar is om te zorgen dat softwareontwikkelingsprojecten een resultaat opleveren dat naar wens en op maat gesneden is voor de gebruiker en opdrachtgever is te kiezen voor een stapsgewijze, iteratieve opzet. Deze benadering staat o.a. bekend onder de namen *Agile, Scrum* of *eXtreme programming.* De ontwikkeling wordt uitgevoerd met korte iteraties, waarbij de klant steeds kan testen of de software doet wat hij/zij verwacht en de functionaliteit bevat die nodig is. Keerzijde van die medaille is dat deze ook een actieve inbreng en een herhaaldelijke inspanning van de gebruikers vergt.

Een middenweg, waarbij fase 1 in drie grotere stappen wordt ingedeeld, kan in dit geval een goede oplossing zijn. De gebruiksvriendelijkheid en functionaliteit van de QC-Wizard kan daarbij geborgd worden door het werk te laten uitvoeren door een inhoudelijk gespecialiseerde softwareontwikkelaar, die allereerst zelf de controles en testen uitvoert. De softwareontwikkelaar doet daarbij allereerst zelf ervaring op met, en wordt ook zelf gedreven door, het efficiënt en gebruiksvriendelijk uitvoeren en verwerken van de protocol-controles. De voorgestelde stappen zijn in concreto:

a) **QC-ImpactTest** – in deze eerste stap wordt 'laaghangend fruit' geplukt, door een aantal eenvoudige en eenduidige controles toe te passen op alle data die beschikbaar is in DINO (liefst van alle meetnetbeheerders) binnen één of meer provincies. Te denken valt aan controle op droogval, overlopen en/of filterverwisseling. Het resultaat geeft antwoord op de vraag in welke ordegrootte het aantal foutieve waarnemingen ligt dat hiermee aan het licht komt (tientallen, honderden, duizenden, of meer?). Dit antwoord geeft daarmee weer zicht op de mogelijke impact en waarde van de QC-Wizard. Elke opgespoorde foutieve waarnemingen voorkomt immers wellicht een terugmelding (per mail?) vanuit de BRO, waarbij de BRO de bronhouder verplicht om daarop te reageren.

b) **QC-Benchmark** – in deze tweede stap wordt hetzelfde principe toegepast, maar dan voor alle controles en richtlijnen uit het protocol, zover dat mogelijk blijkt. Uitvoering van deze stap stelt (veel) meer eisen aan de inhoud en vorm van de beschikbare data (origineel, controle en ruw), de communicatie daarover en verwerking daarvan. Om die reden kan deze stap het beste slechts voor één provincie uitgevoerd, en ook door die provincie apart gefinancierd worden.

c) **QC-Wizard en QC-Workshop v2.x** – De in de vorige stappen ontwikkelde controle-algoritmen en functionaliteit wordt in deze stap omgevormd tot en opgenomen in de gebruikersschil van de QC-Wizard. Het concept-prototype van QC-Wizard v2.x dat hieruit volgt, wordt vervolgens getest en geëvalueerd in een workshop die vergelijkbaar is met de workshop die georganiseerd is rond QC-Wizard v0.5.

4.2 Fase 2/3 : Borging voortgang en ICT-kwaliteit

In fase 1 wordt volgens de voorgestelde aanpak een prototype gerealiseerd dat in operationeel en functioneel opzicht al zo goed mogelijk voldoet. De realisatie van dit prototype brengt als risico met zich mee dat de roep vanuit de gebruiker om een versie die ook in ICT-opzicht voldoende robuustheid en kwaliteit in zich heeft, verstomt. Dit risico kan afdekt worden door een ICT-werkgroep in het leven te roepen, die fase 2 en 3 verder vorm moet gaan geven. De taak van deze ICT-werkgroep is het uitdenken en uitwerken van de

definitieve ICT-architectuur en –inbedding, wat na of zo mogelijk parallel aan de realisatie van het prototype opgepakt kan worden. Ook de vraag hoe en door wie het beheer en onderhoud van de definitieve QC Wizard georganiseerd dienen te worden, dient door de werkgroep beantwoordt en aan het platform voorgelegd te worden

4.3 Risco op een vendor lock-in en scientist lock-out

De keuze van het ontwikkelplatform en de leverancier of uitvoerende partij brengt bepaalde voor- en nadelen met zich mee (zie ook Bijlage A). Een randvoorwaarde bij het realiseren van de QC-Wizard is dat het platform meetnetbeheerders zoveel mogelijk vrij is in de keuze voor een bepaalde leverancier en oplossing bij de eventuele verdere doorontwikkeling en onderhoud van de software. Het voorkomen van een dergelijke *vendor lock-in* is voor een belangrijk deel gewaarborgd door als voorwaarde te stellen dat de QC-Wizard wordt opgeleverd als *open source* en gebruik maakt van open standaarden.

Bij het uitwerken en kiezen van het definitieve ontwikkelplatform, architectuur en onderhoud staat de robuustheid en kwaliteit van de software vanuit ICT-opzicht voorop. Het ligt voor de hand dat daarbij een programmeertaal en ontwikkelplatform gekozen wordt die binnen de ICT meer gangbaar is, waarbij de QC-Wizard door ICT-specialisten gemigreerd zal moeten worden. Dit brengt als risico met zich mee dat de inhoudelijk specialist(en) verder op afstand komen te staan, de overhead toeneemt, het onderhoud bemoeilijkt wordt en het risico op een vanuit inhoudelijk oogpunt suboptimaal resultaat groter is. Een dergelijke zogenaamde *scientist lock-out* kan voorkomen worden door onderdelen die inhoudelijk belangrijk zijn vorm te geven in een programmeertaal die juist onder inhoudelijk specialisten en hydrologen meer gangbaar is (zoals Python of MATLAB).

Literatuur

Von Asmuth, J.R. (2018) Kwaliteitsborging grondwaterstands- en stijghoogtegegevens: Protocol voor datakwaliteitscontrole (QC) (versie 2.0)*;* Rapport PMB2018, Platform meetnetbeheerders grondwaterkwantiteit van de gezamenlijke provincies, Arnhem.

Von Asmuth, J.R. en F.C. Van Geer (2015) Kwaliteitsborging grondwaterstands- en stijghoogtegegevens: Systematiek en methodiek voor datakwaliteitscontrole (QC)*;* Rapportnr. KWR 2015.004 KWR Watercycle Research Institute / TNO, Nieuwegein / Utrecht.

von Asmuth, J.R. en E. Vonk (2017) HydroMonitor - open data exchange format (toelichting en definitie)*;* Report KWR 2016.062, KWR Watercycle Research Institute, Nieuwegein.

Bijlage A: MATLAB als ontwikkelplatform en alternatieven

Inleiding

Alhoewel het voorstel is om de QC-Wizard na realisatie van het prototype te migreren naar een andere ontwikkelplatform, zou dit in de tussentijd toch afhankelijkheid van MATLAB en/of een zekere vorm van *vendor lock* met zich mee kunnen brengen. We gaan in deze bijlage daarom in op de voor- en nadelen, en mogelijkheden van MATLAB, waarbij het vaak gaat om keerzijde van dezelfde medaille. We gaan hier eerst in op de voordelen, die MATLAB voor een groot deel deelt met Python als programmeertaal en ontwikkelplatform, dat in veel opzichten lijkt op MATLAB.

Voordelen

MATLAB is te typeren als commercieel en gedeeltelijk open ontwikkelplatform, dat zich primair richt op ingenieurs en wetenschappers. MATLAB stelt hen in staat om hun ideeën en methoden zo snel en efficiënt mogelijk om te zetten in professionele software. Doordat ingenieurs en wetenschappers dit in grote mate zelfstandig kunnen doen, of daar in ieder geval direct inzicht en invloed in hebben, kunnen dure en langdurige ICT-trajecten worden vermeden. Bij programmeertalen en ontwikkelplatforms die binnen de ICT meer gangbaar zijn moet de software door ICT-specialisten ontwikkeld worden, waardoor de ingenieur per definitie verder op afstand staat, de overhead toeneemt en het risico op een suboptimaal resultaat (vanuit inhoudelijk oogpunt) groter is. De achtergrond hiervan is dat matrix algebra, wiskundige formules en methoden binnen de MATLAB programmeertaal worden opgenomen op een manier die sterk lijkt op de wiskundige notatie zelf. MATLAB code leest dus alsof het de wiskundige formules zelf zijn, wat de code erg compact en inzichtelijk maakt. MATLAB heeft een groep van gebruikers en ontwikkelaars gebruikers die (o.a.) op een eigen manier vrijelijk en onderling MATLAB source code uitwisselen (MATLAB Central).

Zie ook: https://nl.mathworks.com/products/matlab.html

Nadelen en alternatieven

MATLAB is onder ICT-ers een relatief onbekend en weinig gebruikt ontwikkelplatform. Ondanks alle mogelijkheden van MATLAB is er toch een hobbel te nemen wanneer op MATLAB gebaseerde software moet worden opgeschaald en opgenomen in een andere, meer gangbare ICT-omgeving. Omdat ingenieurs en wetenschappers in het algemeen geen gedegen ICT-opleiding hebben gehad, is de groep van bedrijven of partijen die hierin voorziet niet erg groot maar wel aanwezig (zie de *community of developers*).

De Python programmeertaal en ontwikkelplatform lijkt in veel opzichten op MATLAB, maar is niet-commercieel, geheel open source en heeft inmiddels een grote groep van gebruikers en ontwikkelaars. De voor- en nadelen van MATLAB vergeleken met Python zijn in eerste instantie typisch die van commerciële versus niet-commerciële software (betaald versus gratis gebruik, support vanuit een helpdesk of op vrijwillige basis). De ervaring leert dat het gemak en de support en ondersteuning vanuit MATLAB erg goed zijn. Het groeiende aantal ontwikkelaars in Python (ook binnen de hydrologie) maakt echter dat Python steeds aantrekkelijker wordt.

Integratie- en migratiemogelijkheden

Een sterk punt van MATLAB zijn de goede en vele mogelijkheden om MATLAB te vertalen naar of te integreren met andere programmeertalen en applicaties. De integratiemogelijkheden maken het ook mogelijk om de QC-Wizard (naar wens) stapsgewijs en/of gedeeltelijk te migreren. De optie om de QC-database te zien als aparte component of onderdeel, en deze in een ander platform (bijv. DJANGO, zie www.djangoproject.com) te ontwikkelen als web applicatie is al genoemd, maar dit geldt ook voor andere onderdelen van de QC-Wizard.

Omdat Python (in ieder geval binnen de hydrologie) nog weinig gebruikt wordt voor het ontwikkelen van user interfaces en complete applicaties, zijn daarvan zo snel weinig voorbeelden te vinden en zullen de mogelijkheden daarvoor nader bekeken moeten worden.

Web deployment

Bijzondere aandacht gaat uit naar de mogelijkheden om MATLAB applicaties te distribueren en aan te bieden via het internet. MATLAB kent hiervoor verschillende opties:

- MATLAB web apps (https://nl.mathworks.com/help/compiler/web-apps.html)
- MATLAB WebFigures (https://nl.mathworks.com/help/compiler_sdk/java/quick-start-implementing-a-custom-webfigure.html)
- MATLAB Production Server (https://www.mathworks.com/products/matlab-production-server.html)
- MATLAB Online (https://www.mathworks.com/products/matlab-online.html)
- MATLAB in the Cloud (https://www.mathworks.com/cloud.html)

Van deze opties zullen de precieze mogelijkheden ook nader bekeken moeten worden. Voorbeelden van toepassing van de MATLAB Production Server zijn:

Shell: geologists deployed their MATLAB algorithms as an easy-to-use application to share with colleagues worldwide.

NASA: leveraged MATLAB capabilities on multidimensional (engineering) data (like satellite imagery) and deployed an award winning application called: ForWarn.

Community of developers

Alhoewel Python nu de boventoon voert, is ook MATLAB bij ingenieurs, wetenschappers en hydrologen een bekend en breder gebruikt ontwikkelplatform. Voorbeelden van organisaties binnen de watersector die MATLAB gebruiken en applicaties ontwikkelen zijn:

- Deltares (https://www.deltares.nl/nl/)
- TNO (https://www.tno.nl/nl/)
- Royal Haskoning (https://www.royalhaskoningdhv.com/)
- Witteveen & Bos (http://www.witteveenbos.nl/)
- Artesia (http://www.artesia-water.nl/)
- Waterlabs (https://www.waterlabs.eu/)
- AMO (https://www.amo-nl.com/)

Voorbeelden van organisaties met een profiel dat meer gericht is op ICT- en applicatieontwikkeling in het algemeen zijn:

- ModelIT (https://www.modelit.nl/)
- MonkeyProof Solutions (https://monkeyproofsolutions.nl/)

De laatstgenoemde partij is met name interessant, omdat ze zowel MATLAB en Python als meer gangbare programmeertalen als Java, C/C++ en .net gebruiken en aanbieden bij hun

applicatieontwikkeling. Die kennis komt zeker van pas bij het kiezen en migreren van de QC-Wizard naar de definitieve ICT-architectuur.

Conclusie en strategie

Wanneer de code van een MATLAB-applicatie zoals de QC-Wizard open source toegankelijk wordt gemaakt, blijft er sprake van een gedeeltelijke *vendor lock in*, namelijk die met het ontwikkelplatform MATLAB zelf. Alhoewel de afhankelijkheid bij een dergelijke *platform lock in* veel minder sterk is dan die bij een echte *vendor lock in* in het geval van *closed source*, is ze toch ongewenst. De afhankelijkheid en risico's lijken echter goed te beheersen, doordat:

- er een bredere *community of developers* voor MATLAB- applicaties bestaat;
- de QC-Wizard ook stapgewijs en op onderdelen gemigreerd kan worden naar een volledige open source applicatie, waarin eventueel Python de rol van MATLAB overneemt.

De opties en precieze mogelijkheden zullen nader bekeken worden in het vervolgtraject. De mogelijkheden om open source applicaties met behulp van Python te realiseren, zijn ook voor de groep van Python-ontwikkelaars zeker interessant.