Model Builder voor HydroNET

Bachelor Thesis Yorick Fredrix

HydroLogic BV
Postbus 2177
3800 CD Amersfoort
033 4753535
hydrologic.nl

Juni 2016

Voorwoord

Voor u een rapport over het onderzoek voor een model builder in het HydroNET web platform. Deze opdracht is er op gefocust om te onderzoeken of het ontwikkelen van een (flexibele) data analyse module in Python voor HydroNET data zinnig is. Dit onderzoek is uitgevoerd van april tot en met juni 2016.

Binnen het project wil ik een aantal mensen bedanken. Allereerst mijn begeleider bij HydroLogic B.V. Marcel Alderlieste wil ik ontzettend bedanken voor het bieden van een zeer goede begeleiding waarin ik alle ruimte kreeg om keuzes te maken in overleg. Verder wil ik graag HydroLogic B.V. bedanken voor het aanbieden van verbredende opdrachten. Zo heb ik me niet alleen met de bachelor opdracht bezig gehouden, maar toen bleek dat dat sneller dan geplant ging heb ik mee kunnen helpen in andere projecten van het bedrijf, zoals het modeleren van een watersysteem in Sobek en het maken van een wereldwijd verdampingsmodel gebaseerd op Penman-Monteith. De bacheloropdracht is zo ook een zeer leerzame ervaring geweest.
Verder wil ik graag Denie Augustijn bedanken voor het bieden van de nodige ondersteuning vanuit de universiteit Twente vooral op het gebied van schrijven.
Daarnaast wil ik graag mijn mede-stagairs bij HydroLogic B.V. bedanken voor het geven van feedback op mijn werk en het bieden van hulp wanneer dat nodig was.

Amersfoort 1 juli 2016,

Yorick Fredrix

Abstract

Gedurende de thesis is er een onderzoek gedaan naar de mogelijkheden en potentiele voordelen van het ontwikkelen van een model builder voor HydroNET. In de context van de thesis is een model builder een methode om eenvoudig flexibele data analyses te kunnen voltooien met behulp van een python script. Hiertoe is dan ook een framework opgezet die het mogelijk maakt om data te downloaden te analyseren en te uploaden naar een externe locatie. Uit het onderzoek blijkt dat het hebben van een model builder van grote waarde is voor HydroInformatica aangezien het de gebruikers in staat stelt om snel en eenvoudig experimenten te kunnen uit voeren op de data. Met behulp van deze experimenten is het goed mogelijk om tot een beter product ontwikkeling te komen, waarna het goed mogelijk is om alles netjes te analyseren.
Concluderend het prototype van de model builder is zeer succesvol en dat het de moeite waard is om verder te ontwikkelen, tot een product wat beter uitgewerkt is en daarmee bruikbaar voor meerdere mensen.

Inhoud

1 Introductie 2
1.1 Achtergrond en probleem definitie 2
1.2 Doelstelling + onderzoeksvragen 3
1.3 Methodologie 3
1.4 Limieten en grenzen aan het onderzoek 4
1.5 Leeswijzer 5
2 HydroNET 6
2.1 Doel van HydroNET 6
2.2 Gebruik van HydroNET 6
2.3 Mogelijkheden met HydroNET 6
3 Inventarisatie van eisen en wensen voor de model builder van HydroNET 8
3.1 Weather Impact 8
3.2 HydroNET business development 8
3.3 HydroNET ontwikkeling 9
3.4 Model builder overleg 9 mei 9
3.5 Overzicht van alle wensen en eisen 10
4 Prototype model builder van HydroNET 12
4.1 Wat is er al beschikbaar? 12
4.2 Ontwerpkeuzes 12
4.3 Voordelen van de keuzes 13
4.4 Wijze van implementatie 14
4.5 Iteratieslag 16
5 Case studie 18
5.1 Grape Compass 18
5.2 ECMWF Ethiopië 21
6 Evaluatie 24
6.1 Verificatie 24
6.2 Review sessie 24
6.3 Controle werking 25
7 Aanbevelingen 26
7.1 Veiligheid 26
7.2 Foutafhandeling 26
7.3 Data aanvraag limiteren 26
8 Conclusie 28
9 Bibliografie 30
Handige links voor ontwikkeling 30

Bijlage A Programma van Eisen i
Bijlage B Conceptuele data flow iii
B.1 Conceptuele model 1 iii
B.2 Conceptuele model 2 iv
Bijlage C Model Builder uitwerking v
C.1 Versie 1 v
C.2 Versie 2 vii
Bijlage D JSON formaat xi
Bijlage E Grape Compass Case xiii
E.1 Uitwerking Grape Compass xiii
Bijlage F ECMWF Ethiopië case xvii
F.1 Uitwerking in Modelbuilder v1 xvii
F.2 Uitwerking in Model Builder v2 xx
Bijlage G Verificatie xxiii

1 Introductie

1.1 Achtergrond en probleem definitie

Momenteel veranderen er veel dingen op het gebied van waterbeheer door verschillende factoren zoals klimaatverandering, groeiende wereldbevolking, etc. (Brouwer, 2000) Om hier op voorbereid te zijn heeft HydroLogic een software product ontwikkeld genaamd HydroNET. Dit web platform is er op gericht om gebruikers van benodigde informatie op het gebied van water en weer te voorzien om beslissingen te kunnen nemen. Binnen HydroNET zitten een aantal applicaties en functies, wat deze zijn is beschreven in Hoofdstuk 2. (HydroNET BV, 2015)

Aangezien HydroNET een web platform is betekend dit dat je geen installaties of software nodig hebt voor het zien van de informatie enkel een internet verbinding voldoet. Hierdoor is HydroNET ook zeer toegankelijk voor lokale mensen en kunnen bijvoorbeeld waterschappen het gebruiken om zelf beslissingen te nemen alsmede boeren te voorzien van informatie om optimaler met water om te gaan.

Voor de opdracht die er nu ligt is de klant Weather Impact. In een consortium hebben zij een onderzoek gekregen om weersvoorspelling te ontsluiten naar mensen in Ethiopië. In dit geval gaat dan op basis van het ECMWF EPS weermodel. Dit is een ensemble model wat in houdt dat naast gegevens voor locatie en tijd er ook nog variabiliteit in parameters wordt meegenomen. Weather Impact wil behalve een vaste methode voor over de hele wereld in staat zijn om per gebied of in de toekomst zelf eenvoudig andere analyses te gebruiken. (Moltenti, Buizza, Palmer, & Petroliagis, 1994) Op het moment is dit nog niet mogelijk en hier gaat mijn onderzoek op focussen. HydroNET is namelijk heel goed in data analyseren en visualiseren maar wel op een vooraf geprogrammeerde wijze. Daarnaast om dit aan te kunnen passen heb je zeer veel kennis van programmeren nodig wat er niet is bij Weather Impact. Om deze reden willen ze een flexibelere methode om dit mogelijk te maken. Nu is er nog geen idee wat het systeem moet kunnen en hoe het zou moeten werken, dus het eerste punt in de opdracht gaat het bepalen van de wensen en eisen zijn.

Na alle wensen en eisen in kaart gebracht te hebben wordt er een prioritering gemaakt. Deze geeft dan de criteria voor versie 1.0 van de model builder. De model builder is een uitbreiding op HydroNET om flexibele data analyse te ondersteunen en verbeteren. Welke dan ook gecreëerd moet zijn voor het eind van de opdracht. Deze versie 1.0 zou het mogelijk moeten maken om operationeel gebruik te maken en het probleem op te lossen. Natuurlijk moet aan het eind van de versies ook gekeken worden naar een validatie van de uitkomsten. Dat de (kwaliteit van de) uitkomsten van de modelbuilder zijn gewaarborgd. De kracht van de model builder als deze uiteindelijk voltooid is, dat deze makkelijker analyses voor de grote wereldproblemen kan maken. Aangezien het eenvoudiger is om een analyse uit te voeren en databronnen te combineren. Door databronnen te combineren is het mogelijk om meer inzicht te geven in een probleem.

1.2 Doelstelling + onderzoeksvragen

Het doel van dit project is om te bekijken hoe een model builder voor HydroNET vormgegeven moet worden. Om daaruit te komen dienen de volgende onderzoeksvragen beantwoord te worden:

- Wat zijn de eisen aan deze model builder?
- Wat zijn mogelijke vormen voor een HydroNET model builder?
- Wat is de beste vorm voor een HydroNET model builder?
- Hoe presteert de model builder uiteindelijk?

1.3 Methodologie

Om te beginnen is het van belang om alle wensen en eisen voor de model builder te onderzoeken. Om dit te bereiken worden interviews afgenomen met de verschillende afdelingen die belang hebben bij deze model builder. Op de eerste plaats is dit natuurlijk de klant Weather Impact. Echter zijn er meerdere stakeholders voor wie de model builder van belang kan zijn, dit zijn voornamelijk de business development zijde van het bedrijf. Zij verkopen en ondersteunen het product bij klanten over de wereld. Hieruit volgt dus een heel mooi beeld van wat de klanten doen en willen met HydroNET.
Na deze eisen duidelijk in kaart te hebben begint een ontwerpproces, zoals alle ontwerpprocessen gebeurt dit iteratief. Dit houdt in dat er eerst op een conceptueel niveau over een oplossing wordt nagedacht alvorens het geheel uit te werken. Deze conceptuele ideeën zijn besproken met de ontwikkelaars van HydroNET om zo de meest haalbare en toegankelijke optie vanuit mijn achtergrond en die van de gebruikers uit te werken. Deze conceptuele ideeën geven de mogelijke vormen aan voor de model builder.
Dit conceptueel idee wordt daarna ook voorgelegd aan de gebruikers om te controleren of zij de gekozen wijze ook als een werkbare optie zien. Uit deze conceptuele voorstel volgt dan de beste vorm voor de model builder.
Om de prestaties van de model builder te testen wordt er gebruikt gemaakt van case studies. Deze case studies bevatten een aantal aspecten die de wensen voor de model builder duidelijk naar voren laat komen en zich hierdoor laten toetsen of deze informatie in de model builder aanwezig is. Verder wordt er gevalideerd op basis van interviews met gebruikers en de meningen en effecten van deze in kaart gebracht.
Alle onderzoeksvragen worden met een andere methodologie beantwoordt, dit proces is kort samengevat in Fig. 1. De vragen zijn in chronologische volgorde behandeld en er is dus gebruik gemaakt van systeemkunde aanpak. Dit komt erop neer dat er per hoofdstuk ook teruggeblikt wordt op de eisen en gesproken met de opdrachtgever of andere stakeholders om te controleren of het product in de juiste richting ontwikkeld is.
In het totaal zijn er in het project meerdere design iteraties gemaakt. Zo is er naar iedere versie een controle van het resultaat gedaan aan de hand van het programma van eisen. Daarnaast zijn de resultaten doorgesproken met de opdrachtgever.

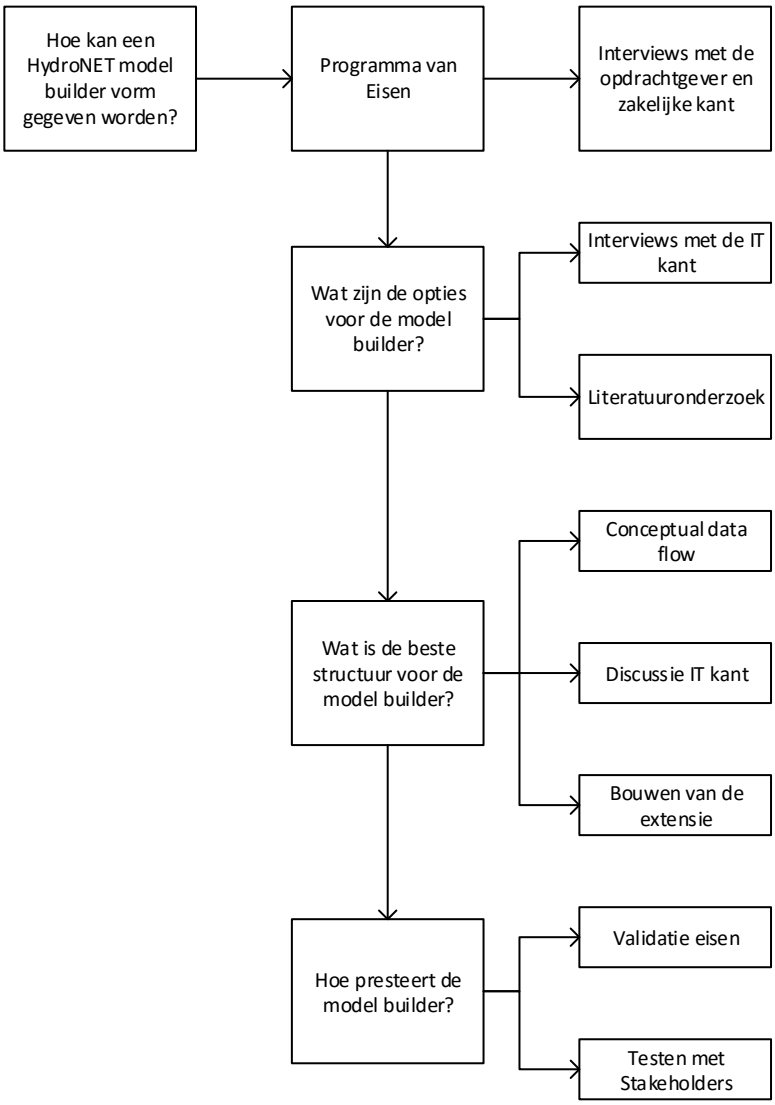

Fig. 1 Methode uitgezet als Schema

1.4 Limieten en grenzen aan het onderzoek

Het onderzoek kent een aantal limieten. Op de allereerste plaatst, is mijn kennis aan het begin van dit onderzoek zeer beperkt op het gebied van python en zeker op HydroNET. Daardoor kan ik geen mogelijke tot de beste model builder komen, maar wel tot een goed prototype. Een andere limitatie is dat het onderzoek zich enkel focust op de nieuwste versie van HydroNET en daarmee tot gevolg dat nog niet alle data en applicaties hierin zitten. Echter is dit wel een meer toekomstgerichte oplossing aangezien op termijn alles naar de nieuwste versie van HydroNET wordt gezet.

Daarnaast is het doel van het onderzoek om te kijken hoe zo'n modelbuilder eruit kan zien en of er een werkend product uit kan komen, dit heeft tot gevolg dat de focus meer op wensen en haalbaarheid gaat liggen dan op de beste oplossing. Daardoor is er uiteindelijk wel een prototype ontwikkeld, maar die hoeft uiteindelijk niet de beste oplossing te zijn. Een andere limitatie is dat het onderzoek zich vooral richt op het vergroten van interne mogelijkheden, hierdoor is er weinig wetenschappelijke basis voor het geheel en dragen enkel de casestudies bij aan wetenschappelijke waarde. Voor de rest zijn de acties vooral gebaseerd op eigen kennis en kennis van het bedrijf.

1.5 Leeswijzer

Allereerst is in hoofdstuk 2 een introductie in HydroNET. Verder zoals zichtbaar in Fig. 1 is de methode zeer gelinkt aan de onderzoeksvragen. Deze worden dan ook beantwoord in verschillende hoofdstukken. Zo gaat hoofdstuk 3 in op het programma van eisen. In hoofdstuk 4 wordt de opbouw van de model builder toegelicht en de werking daarvan. In hoofdstuk 5 worden de gebruikte cases beschreven. Verder volgt in hoofdstuk 6 de evaluatie van het geheel, waar dan ook een terugkoppeling wordt gemaakt naar het programma van eisen. In de hoofdstukken daarna wordt kritisch teruggeblikt op de resultaten en wordt de conclusie nog eens concreet benoemd.

2 HydroNET

2.1 Doel van HydroNET

HydroNET is gecreëerd om waterbeheerders meer inzicht te geven in hun gebied. Dit wordt bijvoorbeeld gedaan door verschillende typen data (waterstanden, gevallen neerslag en weersverwachting) te combineren in één handig overzicht. Hierdoor is het voor waterbeheerders makkelijker om weloverwogen keuzes te maken. Daarmee zijn deze keuzes te verantwoorden naar het publiek. HydroNET levert dus de juiste informatie op een overzichtelijke wijze om beslissingen te kunnen ondersteunen.

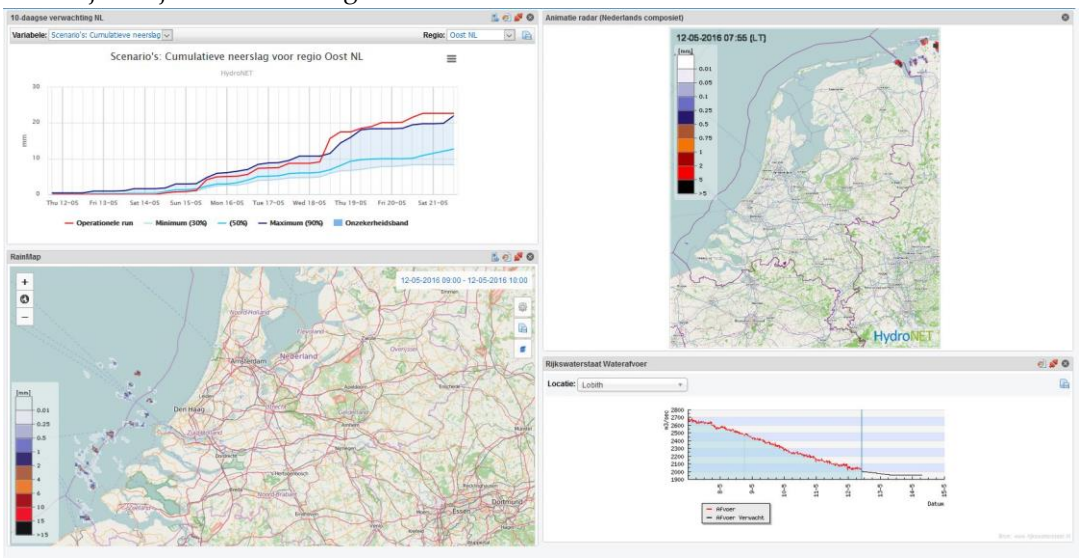

Fig. 2 HydroNET portal dashboard omgeving

2.2 Gebruik van HydroNET

HydroNET wordt over de hele wereld door verschillende bedrijven gebruikt om informatie te verkrijgen. Het HydroNET platform biedt unieke mogelijkheden in het ontsluiten van data. Zo kan er per persoon binnen de organisatie een ander dashboard van dezelfde databronnen gemaakt worden die net andere informatie weergeeft. Hierdoor kan iedereen zijn/haar dashboard zo inrichten als het werk vereist. In het totaal zijn er op het moment meer dan 100 verschillende applicaties die data visualiseren of slimme tools aan koppelen. Onder deze tools vallen zaken als een model of conditioneel formateren. Al deze applicaties geven data op een andere manier weer en helpen zo om de data inzichtelijk te maken voor de gebruiker. In andere woorden HydroNET voorkomt dat mensen verdwaalt raken in een doolhof van data. (Lobbrecht, Einfalt, Reichard, & Poortinga, 2011)

2.3 Mogelijkheden met HydroNET

HydroNET biedt op dit moment tal van mogelijkheden. Zo zijn er meer dan 100 applicaties die hun eigen taak en analyse projecteren voor de gebruiker. Binnen HydroNET zijn veel standaard bewerkingen al geprogrammeerd. In HydroNET kun je met de processors verschillende standaard taken doen, zoals gemiddeldes, maxima, minima etc. Dit soort bewerkingen zijn zaken als projecties wijzigen, gemiddeldes uitreken (bijvoorbeeld per stroom-

gebied of gemeente), standaardafwijkingen enzovoort. Daarbij is het belangrijk om te weten dat nieuwe methodes wel enkel door ervaren programmeurs gemaakt kunnen worden. (HydroNET BV, 2015)
HydroNET is zeer goed in het uitvoeren van vaste taken en het ontsluiten van data. Vele waterschappen gebruiken het om hun eigen data zichtbaar te maken. Hierdoor is het voor het waterschap eenvoudig om allerlei verschillende datasets in dezelfde omgeving te zien. Waarin ook nog databronnen van het KNMI erbij kunnen en dus een waarschuwing voor de beheerders op basis van weersvoorspellingen. Dit kan in de toekomst nog verder gaan door mogelijkheden te creëren dat ook afvoervoorspellingen binnen dat waterschap gemaakt worden op basis van hun eigen data op het moment en deze KNMI voorspellingen. De meest gebruikte functie in HydroNET hierbinnen is het dashboard. Een HydroNET dashboard is niets meer dan een webpagina waar klanten zelf kunnen kiezen welke data ze live willen volgen. Dit is met een “click and drag” technologie in te stellen, de klant kiest dus zelf welke applicaties en databronnen van belang zijn. Deze worden dan voortdurend live weergegeven op hun dashboard. Waterschappen hebben meestal meerder mensen met eigen dashboards, die data tonen op een website of in een app op telefoon of tablet.

3 Inventarisatie van eisen en wensen voor de model builder van HydroNET

Om een duidelijk beeld te krijgen wat de model builder van HydroNET allemaal moet kunnen zijn er interviews gehouden met de betrokken partijen. Deze partijen zijn: “Weather Impact” als opdrachtgever, Business development als stakeholder en IT afdeling als technische ondersteuning. Bij elk van deze partijen is gediscussieerd wat een systeem zou moeten kunnen, waar de limitatie ligt en wat we willen bereiken.

3.1 Weather Impact

Uit de discussie met Weather Impact zijn een aantal belangrijke eisen en wensen naar voren gekomen. Als opdrachtgever hebben zij een praktisch probleem. Dit probleem is dat het momenteel niet mogelijk is om flexibele analyses te maken voor ECMWF data in Ethiopië. Het is namelijk zo dat er iedere dag een model gerund wordt om het weer in Ethiopië te voorspellen. Dit model, het zogenaamde ECMWF model, geeft een voorspel voor de komende 10 dagen in intervallen van 6 uur. Om deze voorspellingen te doen maakt het model gebruik van EPS, EPS is een methode dat 50 simulaties gebruikt. Met deze data is het dan mogelijk om kansen voor weer scenario’s uit te rekenen en dus een voorspelling te maken. Dit dient echter nog gedaan te worden aangezien de data uit het model gewoon 50 tijdsreeksen zijn voor de volgende 10 dagen.
Dit is dan ook meteen het probleem, om deze analyse te vereenvoudigen zou het mooi zijn als dit automatisch zou kunnen gebeuren. Hiertoe willen ze dan ook graag HydroNET uitgebreid zien, dat zij met Python scripts de data standaard automatisch kunnen analyseren. Om dit overzichtelijk te houden zijn de eisen op een rijtje gezet, die zichtbaar is in Bijlage A. De selectie van eisen die meegenomen worden in het prototype zijn te vinden in paragraaf 3.5.

3.2 HydroNET business development

Uit de brainstormsessie met de Business Director Buitenlandse markt: “Leanne Reichard” zijn wensen gekomen die vooral op lange termijn een grote rol spelen. Het idee vanuit de business development is namelijk om de HydroNET te hebben als echt intelligent systeem dat meer doet dan waardes met elkaar te vergelijken of data tonen. Maar in staat is om op grote schaal te vergelijkingen tussen verschillende databases te maken. Hier moet de sterkte van de modelbuilder dan ook vandaan komen. De modelbuilder moet de optie geven, dat mensen met verstand van scripting in staat zijn om verschillende datastromen te kunnen combineren en als nieuwe bron aan te leveren. Een mooi voorbeeld hiervan is de Grape Compass. Het Grape Compass is een toepassing die voor Zuid Afrikaanse wijnboeren de risico’s op schimmels inschat. Dit wordt in detail beschreven in paragraaf 5.1.
Om dit mogelijk te maken is het downloaden en uploaden van de data het meest van belang. Aangezien dit de adaptatie van data en beschikbaarheid van deze adaptie faciliteert. Om te beginnen is het mooi als dit functioneert in een 1 scripting taal, maar naar mate de tijd vordert zou het mooi zijn als meerdere talen ondersteund zouden worden. Daarnaast is het uitvoeren van een onderzoek naar wat is er op het moment al mogelijk zeer gewenst.

Dit om te zien dat het wiel niet 2 keer wordt uitgevonden. Hierdoor zal er dus ook gekeken worden naar wat de mogelijkheden op dit moment al zijn, hoe deze toegepast kunnen worden en wat er al gedaan is.
Uit de brainstormsessie met de Business Director Binnenlandse markt: "Sander Loos" zijn wensen en eisen naar voren gekomen voor zowel korte als lange termijn. Het belangrijkste aspect van de modelbuilder voor HydroNET volgens Sander is dat er een flexibele analyse methode beschikbaar is. Deze methode moet flexibele toetsingen mogelijk maken in HydroNET. Waarna HydroNET deze kan visualiseren. Een voorbeeld hiervoor is de wisselende toetsingsmethode die de Nederlandse Overheid stelt aan chemische stoffen in waterlichamen. De noodzakelijke berekeningen willen nog wel eens veranderen. Zo is het de ene keer concentraties als eis en de andere keer totale hoeveelheid stof. Dat is te programmeren maar is heel veel werk, als het weer veranderd. Hiervoor zou het handig zijn als de klant zelf een script kan creëren die deze complexere analyse kan uitvoeren en kan toetsen. Waarbij dit script gewijzigd kan worden wanneer het nodig is. Hier kan dan natuurlijk een kleurcode via HydroNET aangegeven worden, om het goed mogelijk te maken. Het mooiste is het natuurlijk als deze script afhandeling geïntegreerd zit in HydroNET maar het belangrijkst is dat het een gesloten lus geeft tussen de data ophalen en weer zichtbaar tonen in HydroNET.
In Bijlage A is het programma van eisen zichtbaar met een samengevatte uitkomst van alle gesprekken, in paragraaf 3.5 is te vinden welke eisen worden meegenomen in het prototype.

3.3 HydroNET ontwikkeling

Om de mogelijkheden voor de model builder duidelijk te krijgen is er een interview gehouden met de hoofdontwikkelaar van HydroNET 4 namelijk Jonathan van der Wielen. In dit interview zijn algemene concepten besproken die uitgewerkt kunnen worden tot data flows, zie Bijlage B. Aangezien de ontwikkeling van de model builder geen grote gevolgen heeft voor de ontwikkelingszijde op het moment, waren er niet echt wensen in discussie gekomen. Eerder ging de discussie over de werking van HydroNET en hoe ik daar in kan bijdragen met de ontwikkeling van deze model builder.
Hierin is onder andere besproken dat HydroNET functioneert aan de hand van verschillende C# programma's die onderlinge afhankelijkheden vertonen. Daarmee is dan dus ook duidelijk dat er een verschil is tussen de wensen van Weather Impact en de werking van HydroNET. Aangezien ik geen ervaring heb met programmeren in C# werd mij aangeraden na te denken over oplossingen buiten HydroNET. Om zowel geen last te hebben van dit kennisgat als mede om de processen binnen HydroNET te bewaken en dat er geen plotselinge toename in verbruik plaatsvindt. Dit gesprek heeft geleid tot de meeste ontwerpkeuzes, deze zijn te lezen in paragraaf 4.2.

3.4 Model builder overleg 9 mei

Naast de losse brainstormsessies met individuele personen is er ook een vergadering met alle betrokken personen gehouden om met een ieder te discussiëren over alle ideeën met betrekking tot deze model builder. In deze vergaderingen zijn er allerlei belangrijke punten en vooral verschillen tussen standpunten duidelijk naar voren gekomen. Het doel van de model builder verschilt voornamelijk tussen de ontwikkelaars en de adviseurs. De adviseurs van het bedrijf willen graag meer flexibiliteit in de te maken berekeningen krijgen.

Terwijl de ontwikkellaars graag ontlast willen raken van het implementeren van kleine aanpassingen. Hierdoor is er dus een verschil in aanpak en gedachtegang. Om een visie te hebben is er gezocht naar een gezamenlijke aanpak.
Hieruit kwamen nog een aantal eisen en wensen die zijn opgenomen in Bijlage A.

3.5 Overzicht van alle wensen en eisen

Gezien het doel van het project het onderzoeken of een model builder zinnig is en deze dan als prototype ontwerpen. Is ervoor gekozen om op basis van de interviews hierboven beschreven te beginnen aan een prototype. Dit prototype zal aan de eisen voldoen die te vinden zijn in Tabel 1. Het doel van de eerste versie is om een "proof of concept" te bieden waarmee de sterktes en problemen van een modelbuilder duidelijk naar voren komen. Deze wensen en eisen zijn gekozen om in ieder geval het concept aan te tonen dit wordt gedaan op basis van twee cases, deze worden beschreven in hoofdstuk 5. De wensen en eisen zijn dus zo gekozen dat ze relevant zijn voor de cases, waarmee ze dus ook gecontroleerd kunnen worden.

Tabel 1: Overzicht van alle wensen en eisen die worden meegenomen in de ontwikkeling

Wat	Eis/Wens	Wie?
Model builder gebruikt python scripts	Eis	WI
Model builder heeft goed documentatie	Eis	Iedereen
Model builder kan zelfstandig ECMWF data inlezen en gebruiken	Eis	WI
Model builder is in staat gegevens op te halen	Eis	Business
Werkt met alle data soorten	Wens	Iedereen
Model Builder kan automatisch op dagelijkse interval worden uitgevoerd	Eis	WI
Analyse moet live draaien of periodiek	Wens	Business
Complexere analyses mogelijk maken	Eis	Bedrijfsvisie
Model builder draait scripts op geplande momenten	Wens	Business
Model builder functioneert buiten HydroNET (ook bronnen buiten HydroNET)	Wens	Potentiële gebruiker
Model builder is bruikbaar intern en partners	Wens	Bedrijfsvisie
Model builder is in staat gegevens te bewerken	Eis	Business
Modelbuilder maakt analyses beheersbaar	Eis	Potentiële gebruiker
Model builder kan ook andere data inlezen op keuze van de gebruiker	Wens	WI
Model builder moet de uitkomsten op een ftp server kwijt kunnen	Wens	WI
Gebaseerd op bestaande ideeën	Optie	Business

4 Prototype model builder van HydroNET

4.1 Wat is er al beschikbaar?

Om de data te analyseren binnen HydroNET zijn er aan aantal opties die bestaan. Allereerst is het mogelijk om binnen HydroNET een processor te bouwen. Een processor is een tool die berekeningen kan maken vanuit de datasets binnen de server. Deze processoren werken in real time, dit betekent dat bij de aanvraag van een dataset deze berekend wordt door de processoren. Een alternatief hiervoor is het van te voren uitrekenen van mogelijke aanvragen en deze dan tonen. Dit heeft als nadeel dat het niet flexibel is in de keuze van de data. HydroNET werkt met real time processing voor een groot deel, wat als voordeel heeft dat de data beschikbaar is op aanvraag en je dus flexibeler bent.
Om nieuwe procesoren binnen HydroNET te maken heb je een ervaren programmeur C# nodig. Aangezien deze groep binnen HydroLogic beperkt is, maakt het de ontwikkeling van deze analyse tools tot een langzaam proces. Om dit te vereenvoudigen komt mijn onderzoek in beeld. Het idee van mijn onderzoek is om het mogelijk te maken analyses die misschien over de tijd dienen te wijzigen te faciliteren zonder dat daar een ervaren programmeur aan te pas komt. Om dit mogelijk te maken zijn er een aantal belangrijke punten namelijk het verkrijgen van de data en het kunnen tonen van de uitkomsten. Op beide gebieden is al wat beschikbaar vanuit het bedrijf.
Voor het verkrijgen van data is een API (Application Programming Interface) ontwikkeld die het mogelijk maakt om data aanvragen te doen aan de server. Deze aanvragen worden dan weergegeven in een JSON formaat. Dit is goed te gebruiken aangezien JSON een algemeen formaat is wat over meerder programmeertalen te gebruiken valt. Het nadeel aan JSON is dat het niet de meest efficiënte formaat is om data te versturen voor een computer. Het voordeel daaraan is wel dat de data eenvoudig te lezen en interpreteren valt voor mensen.
Binnen in de API zitten ook allerlei functies om de output te converteren naar andere bestandsformaten zoals csv, xlsx, xml, enz. Dit geeft de flexibiliteit om data op te kunnen vragen en in verschillende programma's te analyseren is.
Daarnaast is het laatst mogelijk geworden om ArcGIS te gebruiken, met de data van HydroNET. Hierdoor kun je vele ruimtelijke analyses uitvoeren op de data zolang je een ArcGIS licentie hebt.
In andere woorden er is al heel veel mogelijk echter niet alles. Zo is het zeer moeilijk om te experimenteren met applicatie ideeën, terwijl dat heel belangrijk is voor de ontwikkeling van nieuwe applicaties. Daarnaast is het heel lastig om allerlei verschillende datasets te combineren in HydroNET. De modelbuilder maakt het mogelijk om gemakkelijk data te verkrijgen op een gestandaardiseerde methode. Daarna zorgt de modelbuilder ervoor dat er op een eenvoudige manier complexere analyses op de data uit te voeren. Het idee is dat de modelbuilder het mogelijk maakt om als niet ervaren programmeur toch een analyse te kunnen doen.

4.2 Ontwerpkeuzes

Aangezien niet alle ontwerppunten vast liggen in de eisen zijn er keuzes gemaakt op basis van documentatie en gesprekken met de ontwikkelingsafdeling. Deze keuzes worden hier toegelicht, in paragraaf 4.3 staan de voordelen van deze keuzes.

In het proces om deze model builder te creëren zijn allerlei keuzes gemaakt. De allereerste keuze was de taal waarin de analyses uitgevoerd moesten worden. Aangezien een wens van de opdrachtgever Weather Impact is dat de analyses in Python functioneren was deze keuze snel gemaakt. Daarna kwam dan de vraag hoe we Python gingen gebruiken. Hiervoor zijn 2 conceptontwerpen gemaakt zie Bijlage B. Hieruit is gekozen dat de model builder los van HydroNET gaat functioneren. Om 2 redenen allereerst zodat de implementatie eenvoudiger is aangezien je geen rekening hoeft te houden met C# klassen, maar enkel een goed gedocumenteerde API. Daarnaast zorgt het er ook voor dat geen bronnen van HydroNET gebruikt worden voor de analyses, zie paragraaf 3.3. (Van Der Wielen, 2015)

De API van HydroNET 4 gebruikt een post request om de data op te vragen, dit heeft als voordeel dat resultaten van bekende berekeningen ook opgevraagd kunnen worden zonder dat deze van te voren berekend hoeven te zijn. (Fielding, et al., 1999). Waarna hier een respons met data opgestuurd wordt in een JSON formaat. JSON is een formaat dat is opgebouwd uit hoofdstukken, zo heb je dus een kopje data met daarbinnen per dag een tab waar de data achter ligt bijvoorbeeld. Daarnaast biedt JSON verscheidene opties om eenvoudig inzage in de data te hebben en is het een universeel data formaat waardoor bijna alle talen het ondersteunen.
Na beraad met de opdrachtgever (Weather Impact) is besloten om de functionaliteit in te bouwen om data (i.e. het resultaat van een analyse) naar een ftp server weg te schrijven. Deze data dient aangeleverd te worden in een specifiek JSON formaat. Dit aangezien de afspraak met de klant van Weather Impact is gemaakt om de data in een bepaalde JSON te leveren. Om dit mogelijk te maken is er gekozen voor 2 extra modules aan de module builder toe te voegen. Deze modules zijn een FTP module en een JSON string maak module.
De JSON module: De JSON module bouwt van de gegeven informatie een JSON string met de afgesproken opbouw zoals gewenst bij de opdrachtgever. Dit formaat wordt dan generiek gecreëerd op basis van een naam en de gekozen data. Om het formaat te controleren zijn de resultaten van de JSON module besproken met de opdrachtgever. Uit deze gesprekken kwamen ideeën voor veranderingen maar die zijn besproken als niet flexibel integreerbaar. Hierdoor is gekozen om deze niet verder door te voeren en het formaat zo te houden.
De FTP module is opgebouwd uit een generieke functie. Hier is voor gekozen omdat deze dan eenvoudig te gebruiken blijft voor alle momenten in de toekomst. Dit begint nu bij de huidig vereiste server, maar kan dan ook veranderen naar een andere server. Daarnaast biedt het de optie om eenvoudig de FTP server te bereiken.

4.3 Voordelen van de keuzes

De modelbuilder is als een losse module/applicatie gebouwd. Dit heeft dat een aantal grote voordelen in combinatie met Python. De losse identiteit zorgt ervoor dat je enigszins onafhankelijk bent van HydroNET en zodoende het niet uit maakt hoe zwaar de analyses zijn die je wilt draaien, en dat je deze op elke pc kan uitvoeren. Een van de voordelen dat de model builder buiten HydroNET draait in python is dat Python is open source en cross platform. Open source zorgt ervoor dat het eenvoudiger is om meer functies te maken in de vorm van libraries, daarnaast ook nog eens dat de software gratis is. Cross platform en

buiten HydroNET zorgt ervoor dat de modelbuilder op alle operation systemen werkt. Python heeft als open source script taal het voordeel dat allerlei mensen er al functies voor geschreven hebben. Deze functies zijn verpakt in een library die als voordeel heeft uitgebreid gedocumenteerd te zijn en daardoor relatief eenvoudig in gebruik. Zo is het mogelijk om met de model builder actief waarschuwingen te versturen naar gewenste partijen. Gezien de mogelijkheden in python kan dit zowel via e-mail als sms bijvoorbeeld. Dit is maar een van de voorbeelden die aantonen wat de flexibiliteit dankzij python. Daarnaast heeft python het grote voordeel dat er ook ruimtelijke analyse mogelijk is. Hiermee zou een grid kunnen worden gesneden met een polygoon, waardoor bijvoorbeeld de neerslag per gemeente kan worden uitgerekend.

4.4 Wijze van implementatie

De keuzes beschreven in paragraaf 4.2 en het gekozen conceptuele model in B.1 zijn uitgewerkt in het prototype. Dit is gedaan in Python, zoals eerder genoemd. Binnen Python wordt gebruik gemaakt van een aantal libraries die het mogelijk maken om het gekozen model volgens het schema zichtbaar in Fig. 3 uit te werken. Dit is dus een modulaire structuur, waarbij het eerste deel is de download module, daarna volgt een mogelijke analyse en een upload module. Dit maakt het mogelijk om snel de analyse aan te passen, of extra analyses toe te voegen.

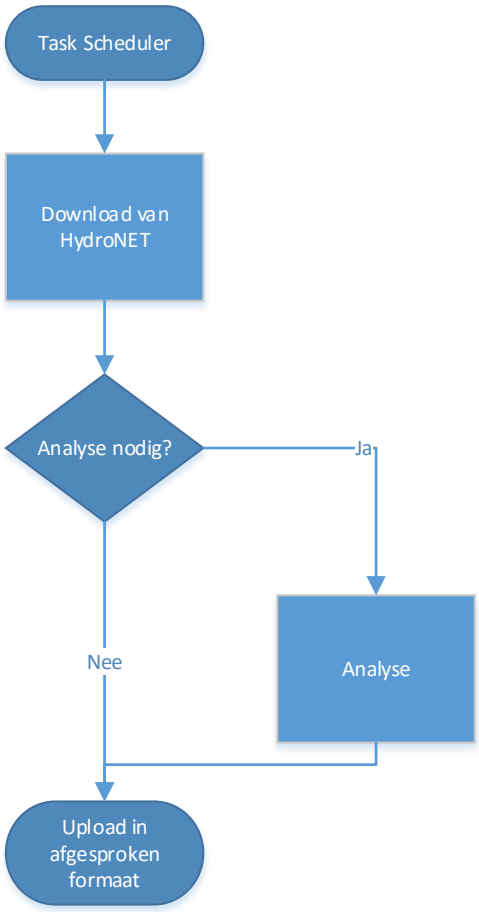

Fig. 3 Werking van de Model Builder

4.4.1 Download module

De download module is opgezet als de eerste stap in het proces, in deze module krijgt de gebruiker alle vrijheid om alle mogelijke databronnen op te vragen in HydroNET (grids, modelgrids, tijdreeksen etc.). Daarnaast is het ook mogelijk om bij deze aanvraag een processor toe te voegen. Een processor is een tool binnen HydroNET om een aantal standaard bewerkingen uit te voeren, zie paragraaf 2.3. Hiermee is het mogelijk om alvast het 90ste percentiel uit te rekenen van de ECMWF dataset. Daardoor krijg je dan dus niet meer alle ensembles terug maar enkel het 90ste percentiel. Hierbij wordt het mogelijk om deze op te slaan in de data klasse die aangemaakt wordt. Binnen die klasse wordt alle noodzakelijk informatie per sub punt gemaakt Een van die sub punten bevat de exacte HydroNET respons. Zodat mocht er in de toekomst nog andere meta informatie nodig is deze daaruit gehaald kan worden. Op het moment is de 10 belangrijke metadata, namelijk het grid, starttijden, eindtijden, interval, variabelen en eenheden.

4.4.2 Analyse

Na het succesvol doorlopen van de download module is de data gereed en kunnen er analyses/bewerkingen uitgevoerd worden op de data welke nog niet uitgevoerd zijn in HydroNET. Daarnaast is het hier mogelijk om daadwerkelijk alles wat in Python mogelijk is met je data te kunnen doen. De enige beperking daarbinnen is dat de data is opgeslagen in een numpy array en je zal dus vanuit die startpositie de analyse en het resultaat moeten leveren. Ondanks deze beperking laat dat nog tal van mogelijkheden open, zoals het uitrekenen van een neerslagtekort, het uitvoeren van tijdsreeksanalyses, gridcorrecties, etc. Een aantal mogelijke tijdsreeksanalyse zijn regressie analyse, autocorrelatie, ARIMA modellen, voorspellen, extrapoleren, interpoleren.

4.4.3 Uploaden

Nadat de gebruiker de correcte data hebt opgehaald en begonnen is met de analyse, is het van belang om het resultaat te tonen en op te slaan in het formaat dat is afgesproken met de opdrachtgever. Dit formaat is een JSON (Javascript Object Notation) string waarin de data zit per starttijd. Daarnaast nog zijn er de algemene metadata, zoals de grid definitie, beschikbare variabelen en de eenheden. Het bestand ziet er dan uit zoals zichtbaar in Bijlage D.
Daarna is besproken dat de data (in JSON formaat) beschikbaar moet worden gemaakt op een FTP server. De klanten van Weather Impact kunnen vervolgens de resultaten van de analyse gemakkelijk vanaf de FTP downloaden. Een stukje python code van deze upload is te zien in Fig. 4. Dit stukje code geeft aan dat er automatisch gezorgd wordt, dat alle data die nodig is gekopieerd wordt naar de ftp server. Hiertoe wordt eerst een lokale kopie opgeslagen die wordt geüpload naar de ftp waarna de lokale kopie wordt verwijderd. De ftp

```
for i in range(len(filename)):
  with open(filename[i],'w') as A:
    json.dump(data[i],A)
    ftp.storbinary("STOR "+filename[i],open(filename[i],'rb'))
    A.close()
  os.remove(filename[i])
```

Fig. 4 Stukje python code van FTP upload

module is zo opgebouwd dat deze werkt voor alle verschillende soorten FTP servers. Het is te allen tijde mogelijk om data naar de FTP server te sturen; dat wil zeggen dat zowel de bron data (beschikbaar na de download module) als allerlei analyses naar de FTP kunnen worden weggeschreven.

4.4.4 Overige hulpmodules

Daarnaast is er een module geschreven die het mogelijk maakt om tijd en datums om te zetten naar een string in plaats van een date time formaat. Dit is noodzakelijk om aanvragen te kunnen doen aan de server en om datums op te slaan in het JSON formaat. Er zijn daar een paar kleine puntjes om rekening mee te houden vooral dat maanden, dagen, uren en minuten altijd uit 2 getallen moeten bestaan, dus voor minder dan 10 moet er een nul toegevoegd worden.

4.4.5 Foutafhandeling

Om ervoor te zorgen dat de gebruiker op de hoogte is van alle fouten die plaats vinden in de model builder, zowel in ontwikkeling als operationeel gebruik. Op de plekken waar dingen fout kunnen gaan zit een functie ingebouwd die een melding maakt waar de fout vandaan komt en hoe die op te lossen is. Naast deze melding wordt ook een mail verstuurd met dezelfde informatie verstuurd naar een mail adres naar keuze. De punten in de model builder waar deze error meldingen kunnen komen zijn: Verkeerde type bij de databron opgevraagd, datum bestaat niet, download fout, interval kan niet en er is geen data beschikbaar. De mail stelt de gebruiker en of klant op de hoogte dat de informatie die er is niet hoeft te kloppen. Verder zorgt de e-mail ervoor dat je als gebruiker in operationeel gebruik op de hoogte gesteld wordt als er een fout ontstaat.
Deze module heeft nog meer opties, dan enkel foutafhandeling. Zo kan de functie om mails te versturen ook gebruikt worden om actief te waarschuwen voor mogelijk extreme situaties. Mocht er dus in je analyse een belangrijk resultaat uit komen, bijvoorbeeld een extreme hoeveelheid neerslag, dan ben je in staat om de mail te versturen naar de gebruiker met een actieve waarschuwing. Dit heeft dus als effect dat mensen eerder op de hoogte zijn van situaties en ze zich hier beter op kunnen voorbereiden.

4.5 Iteratieslag

Na de eerste versie online en werkbaar te hebben is er gereflecteerd met de klant. Allereerst was de conclusie van het gesprek dat de opdracht was voldaan. Alleen dat er nog mogelijkheden waren om de methode te verbeteren. De eerste versie van de modelbuilder was namelijk expliciet geprogrammeerd. Daarmee doel ik op alle losse componenten in losse variabelen te schrijven en alles uitwerken zodat het zeer duidelijk is voor jezelf wat er gebeurt zonder het overzicht te verliezen. Het gevolg hiervan dat je zeer eenvoudig een typefout kon maken aangezien het ophalen van data, alleen al 11 variabelen als output geeft. Op basis van dit probleem heb ik onderzocht of het gebruik maken van klasse gebaseerd programmeren dit kon vereenvoudigen. De conclusie hierop is ja, klasse gebaseerd programmeren maakt het namelijk mogelijk om de 11 argumenten in een klasse te stoppen waarmee typfouten ernstig worden gereduceerd. (Anderson & Drossopoulou, 2004) Het nadeel is alleen dat je nu minder duidelijk ziet hoe de code functioneert. Deze verschillen

zijn te vinden in de python code van de modelbuilder zie Bijlage C. Daarbinnen is zichtbaar wat de verschillen zijn tussen versie 1 en versie 2. Daarnaast zijn de gebruikersverschillen zichtbaar in Bijlage F en worden deze toegelicht in paragraaf 5.2.

5 Case studie

Uit de gesprekken beschreven in hoofdstuk 3 zijn drie duidelijke cases naar boven gekomen. Daaruit worden de volgende uitgewerkt voor de model builder. Allereerst het analyseren van de ECMWF data voor Ethiopië en op de tweede plaats het namaken van het Grape Compass. De ECMWF data is bedoeld voor operationeel gebruik hiermee wordt dus vrij veel van de model builder verlangt, maar heeft niet echt validerende waarde. De case van het Grape Compass is meer als validatie bedoeld. Het Grape is op het moment al operationeel in gebruik in HydroNET en als de modelbuilder dus op dezelfde resultaten uitkomt dan blijkt deze te werken. Om deze reden is dan ook eerst het Grape Compass uitgewerkt om aan te tonen dat het systeem werkt en daarna is de ECMWF data bekeken.

5.1 Grape Compass

Grape Compass is een applicatie ontwikkelt door HydroLogic voor wijnboeren in zuid Afrika. Grape Compass voorspelt de kans op schimmels in de druivenranken in de Stellenbosch regio in Zuid-Afrika. Momenteel wordt er om schimmels te voorkomen preventief gesproeid door de boeren. Dit wordt gedaan door standaard eens per 2 weken alle druiven met pesticiden te voorzien. Dit heeft een grote inpak op het milieu en kost natuurlijk veel geld. Door slim de schimmels te voorspellen kun je het sproeien van pesticiden uit stellen. Het heeft natuurlijk geen zin om te sproeien als de omstandigheden van de natuur zo zijn dat er geen schimmels groeien. Aangezien de schimmels een enorme invloed hebben op de groei en kwaliteit van de druiven, dus ook de wijn. Is het van belang dat alle vormen van schimmel gecheckt worden. Daarvoor wordt de voorspelling gedaan op basis van drie schimmelmodellen. Deze modellen zijn de Downy mildew, Powdery Mildew en Botrytis schimmel modellen, deze schimmels zijn te zien in respectievelijk Fig. 5, Fig. 6 en Fig. 7. Deze modellen berekenen de kans op schimmelgroei op basis van een aantal variabelen, namelijk “Leaf Wettness Duration”, Temperatuur en neerslag.

Fig. 5: Downy Mildew

Fig. 6: Powdery Mildew

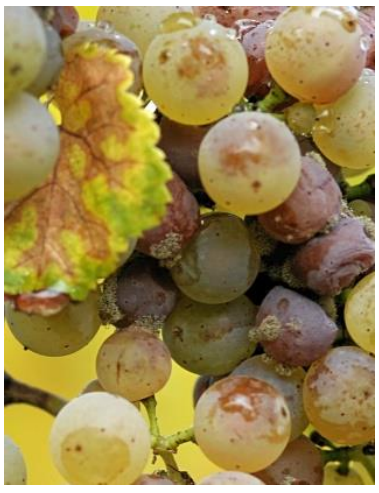

Fig. 7: Botrytis

De variabelen nodig voor de schimmelmodellen worden geleverd door het ECMWF model (uitleg zie hoofdstuk 5.2) vanuit het KNMI voor enkel de Stellenbos regio in Zuid Afrika. Hierin zitten alle variabelen behalve de "Leaf Wettness Duration", deze is echter af te leiden van de relatieve luchtvochtigheid. Aangezien de relatieve luchtvochtigheid ook niet wordt meegeleverd dient deze eerst bepaald te worden aan de hand van de temperatuur en de dauwpunt temperatuur. Hiermee is het dus mogelijk om de parameters voor de schimmelmodellen te bepalen uit één databron.
Aangezien deze case als validatie gebruikt wordt van de model builder is ervoor gekozen om enkel het Botrytis model uit te werken. De aanname wordt gedaan dat als één model werkt is de werking van de modelbuilder aangetoond. Daarbij is ook gekeken of de tussen variabelen al gelijk zijn in plaats van enkel het eindresultaat. Hierover wordt meer toegelicht in hoofdstuk 5.1.2.

5.1.1 Uitwerking Grape Compass

Er is dus besloten om de Grape Compass maar voor 1 type schimmel uit te werken deze schimmel is geworden de Botrytis. Dit heeft te maken met de eenvoud van het schimmel model. Dit maakt het makkelijker om verschillen te detecteren en te verklaren. Hierdoor is dit model het meest geschikt voor de validatie van de modelbuilder.
Het Botrytis model is als volgt:

$$II = -2{,}647866 - 0{,}374927 * LWD + 0{,}061601 * LWD * T - 0{,}001511 * LWD * T^2$$
$$Met\ T = 12°C\ als\ T < 12°C$$
$$Met\ T = 32°C\ als\ 32°C < T < 40°C$$
$$Met\ II \leq 0\ als\ T > 40°C$$

In deze formule is II de infectie index, LWD de "Leaf Wettness Duration" en T de temperatuur. Daarbij is het dan nog van belang hoe de LWD wordt bepaald. In principe is de LWD niks anders dan het aantal uren dat de relatieve luchtvochtigheid groter is dan 90%, zonder voor ten minste 3 uur daar onder te vallen. Echter is de ECMWF brondata enkel beschikbaar per 3 of 6 uur. Om dit dan dus te bepalen is het simpelweg de tijd dat de gemiddelde relatieve luchtvochtigheid groter blijft dan 90%. (Broom, English, Marois, Latorre, & Aviles, 1995)

De relatieve luchtvochtigheid is te bepalen aan de hand van de dampdruk in de lucht:

$$RH = \frac{e_a}{e_0} * 100\% = \frac{0{,}6108 * e^{\frac{17{,}27 * T_{dauw}}{T_{dauw}+237{,}3}}}{0{,}6108 * e^{\frac{17{,}27 * T_{gem}}{T_{gem}+237{,}3}}}$$

Hierin is RH de relatieve luchtvochtigheid en T_{dauw} de dauwpunt temperatuur en T_{gem} de gemiddelde temperatuur. Deze temperaturen zijn beschikbaar in de ECMWF dataset en kunnen dus gewoon gedownload worden. Voor de uitwerking van deze case in code staat verder toegelicht en uitgewerkt in Bijlage D.

5.1.2 Resultaten Grape Compass

Na de implementatie in python is de berekening uitgevoerd. Dit python script creëert CSV files die het grid tonen met de waardes per cel. Naast deze data van de modelbuilder is de brondata (netCDF) uit HydroNET gebruikt om de Leaf Wettness Duration (LWD) te vergelijken. Door deze twee bestanden met elkaar te vergelijken is het mogelijk om iets te zeggen het proces van de modelbuilder te valideren.

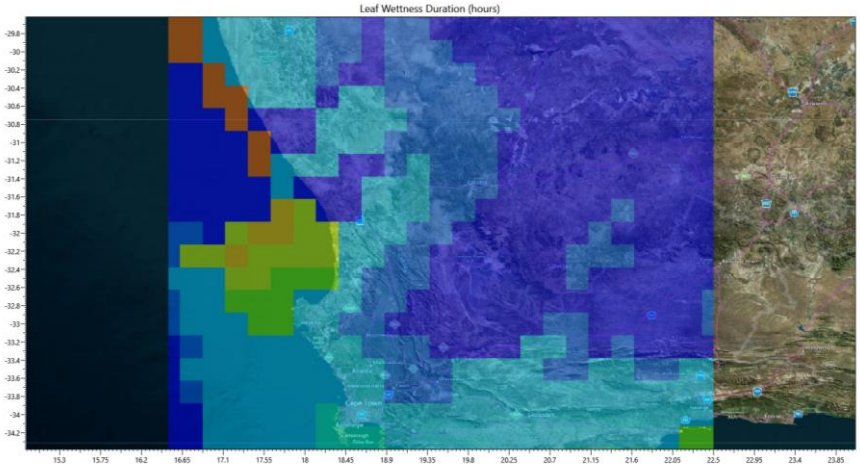

Fig. 8: Leaf Wettness Duration van HydroNET van 18-02-2016 18:00-24:00

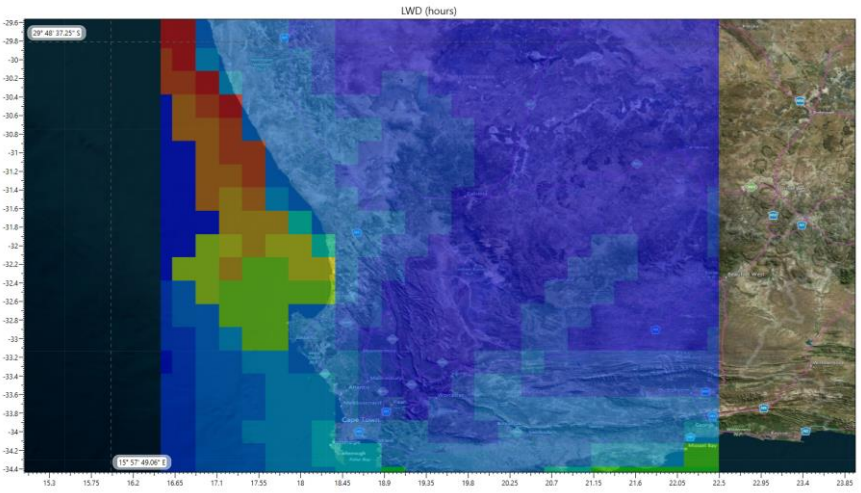

Fig. 9: Leaf Wettness Duration Modelbuilder van 18-02-2016 21:00-24:00

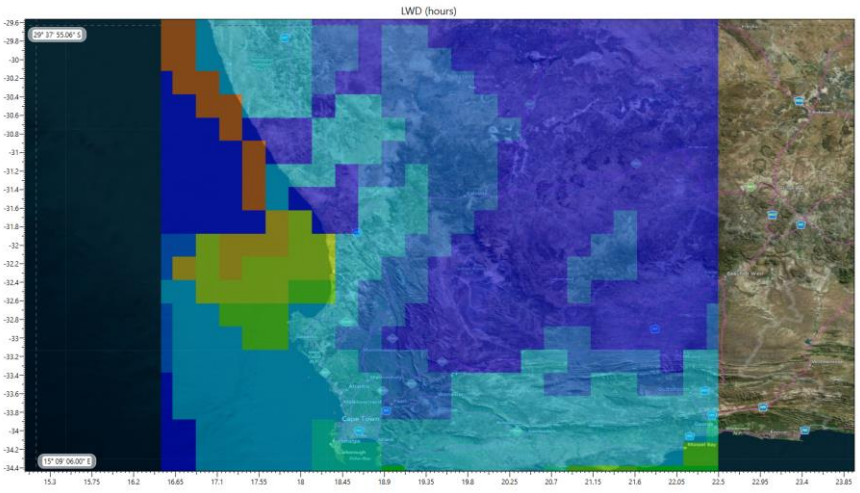

Fig. 10: Leaf Wettness Duration Modelbuilder van 18-02-2016 18:00-24:00

Door Fig. 8 en Fig. 10 met elkaar te vergelijken is zichtbaar dat de vorm van de grote ongeveer hetzelfde is. Dat toont aan dat het proces van de model builder werkt. Kleine verschillen tussen enkele grid cellen ontstaan doordat dat de berekeningsmethode van het 90ste percentiel in HydroNET selecteren niet bekend is en daar dus een eigen approach voor is

ontwikkeld. In de model builder wordt er eerst alle luchtvochtigheden uitgerekend en daar het 90ste percentiel van genomen. Dit heeft tot gevolg dat mocht dit in HydroNET anders gebeuren er iets andere waardes uitkomen.
Daarnaast door Fig. 8 en Fig. 9 met elkaar te vergelijken is het effect van de tijdsschaal op de resultaten zichtbaar. Dit is gedaan omdat er ook gekeken moest worden van het bedrijf of op een kortere tijdsinterval rekenen nauwkeurige resultaten op levert en dus vaker een positieve uitslag geeft. Dit is van belang omdat de LWD de grootste invloed heeft op de kans op schimmels.
Aangezien de vorm van het figuur met de hoge waardes (rood) en de lage waardes (blauw) hetzelfde patroon vertoond in Fig. 8, Fig. 9 en Fig. 10 moet de modelbuilder correct zijn taken uitvoeren. Daarbij wordt dus de juiste temperatuur ingeladen en wordt de correcte luchtvochtigheid uitgerekend. Echter maakt het tijdsinterval verschil het moeilijk om goede uitspraken te doen over de exacte werking.
Ondanks deze verschillen is het toch een duidelijkheid dat de modelbuilder functioneert naar de behoren. Daarnaast toont de proef de kracht van de modelbuilder aan, aangezien de berekening eenvoudig van interval kan variëren door een enkele parameter te wijzigen. Daardoor is de model builder veel flexibeler dan de codes in HydroNET waar dit allemaal hardcoded in zit. Verder is in de model builder eenvoudig een extra schimmelmodel toe te voegen, door enkel een extra formule erin te zetten, terwijl dit in HydroNET aan allerlei andere datasets gekoppeld dient te worden. Dit maakt het voor mensen zonder veel programeer kennis zeer moeilijk zo niet bijna onmogelijk om even een extra model toe te voegen.
Verder is het nog mogelijk om met de model builder de lokale boer actief te waarschuwen voor schimmel op zijn veld. Doordat je in de model builder een mail/sms kan versturen naar de lokale boer als de kans op schimmel in zijn gebied te groot wordt. Hiermee is het een grote toevoeging voor HydroNET wat in de toekomst nog grote potentie biedt.

5.2 ECMWF Ethiopië

ECMWF staat voor “European Centre for Midrange Weather Forecasts” oftewel het Europese centrum voor middellange termijn weersvoorspellingen. In het jaar 1992 is er door hun een model ontwikkeld om wereldwijd weersvoorspelling te kunnen maken op een grid van 0,1° dit is gelijk aan ongeveer 9*9 km. Dit is de operationele run in een deterministische aanpak, daarnaast is er ook nog een ensemble prediction van het ECMWF. Dit wordt gecreëerd op een grid van 0,2° oftewel 18*18 km. Deze ensemble predicition is bedoeld om een schatting van de onzekerheid te kunnen maken in de deterministische voorspelling. De ensemble prediction van de ECMWF bestaat uit 50 ensembles en 1 controle run. Deze controle run is aanwezig om te tonen wat het effect van een andere grid heeft op de berekening. (Moltenti, Buizza, Palmer, & Petroliagis, 1994) Gedetailleerde informatie over ECMWF en analyses met ECMWF worden toegelicht in paragraaf 5.2.2 en in Bijlage F.

5.2.1 Vraagstuk opdrachtgever

Zoals in paragraaf 3.1 al is toegelicht wil de opdrachtgever flexibel analyses kunnen doen op de ECMWF data die in HydroNET beschikbaar is. Deze data kan binnen HydroNET geanalyseerd worden mits je gedetailleerde programmeerkennis hebt op het gebied van C#. Aangezien dit binnen Weather Impact niet het geval is, dienen dit soort klussen altijd te

worden gevraagd aan een HydroLogic collega met C# kennis, en dat maakt het proces lastig en traag.
Het idee is dus dat er een platform moet ontstaan waarin eenvoudig analyses te voltooien zijn. Om aan te tonen dat de modelbuilder dat goed kan is er dus een selectie binnen deze analyse gemaakt door mij als "proof of concept". (Van Der Burgt & Van Pelt, 2016)

5.2.2 Uitwerking

Binnen de case is gekozen om een aantal punten uit te werken, namelijk:

- Maximum temperatuur
- Minimum temperatuur
- Windstoten
- Kans op neerslag
- Minimum neerslag
- Maximum neerslag

Dit is gedaan aangezien deze punten het belangrijkst zijn voor de klant van de opdrachtgever en het voor een "proof of concept" voldoende is. Deze punten zijn uitgewerkt op basis van een aantal analyse acties. Sommige van deze analyse acties gebeuren binnen python andere worden uitgevoerd in de HydroNET omgeving.
In dit geval wordt alles wat al in HydroNET mogelijk is ook met behulp van HydroNET procesoren gebruik gemaakt. Dat heeft het voordeel dat je zowel de sterktes van HydroNET als Python kan gebruiken. In deze case worden dan dus de ook de maxima en minima uitgerekend door HydroNET. Dit is van belang op de temperatuur en de wind om aan de gevraagde parameters te komen. Daarbij is er nu enkel gevraagd om een voorspelling waarvoor enkel het nauwkeurige deterministische model wordt gebruikt hierdoor kan er geen schatting van de onzekerheid gegeven worden.

```
Pminpercent=30/100.
minind=int(np.floor((np.shape(forecastP1[0,1:50,:,:,:])[0])*Pminpercent))
forecast_pmin1=np.sort(forecastP1[0,1:50,:,:,:],0)[minind]
```

Fig. 11: Python implementatie minimum neerslaghoeveelheid

Aangezien bij de neerslag de kans van belang is wordt er gebruikt gemaakt van de ensemble data van de ECMWF. Deze data is beschikbaar in HydroNET en kan worden opgevraagd. Daarna worden alle mogelijke analyses in HydroNET uitgevoerd. Dit bevat niet alles dus de overige analyses worden zoals bedoelt in Python gedaan. De ensemble van de EMCWF is een EPS (Ensemble Predicition system). Het EPS van de ECMWF bevat 50 ensembles waaronder ook alle extremen die meestal onrealistisch zijn. Daarvoor wordt er dus gekozen om de ensembles te beperken tot 30% en 90%. Daarbij is 30% is dan het minimum en 90% het maximum van de voorspelling. Dit wordt dus voor de neerslag bepaald. Om op 30% te komen wordt per cel alle waardes gesorteerd, waarna hiervan dan de 15de ensemble gekozen wordt. Dit wordt in python uitgevoerd door de correcte indices van ensembles te kiezen met de bijbehorende kans, zie Fig. 11.
De andere berekening met de neerslag namelijk de kans op neerslag wordt ook in python berekend. Dit gebeurt door over alle cellen heen te loopen in zowel tijd als locatie en te controleren over er meer neerslag valt dan de geplaatste grenswaarde van 0,5mm. Als dit het geval is wordt dit is als een neerslag scenario beschouwt waarna het aantal neerslag

scenario's gedeeld wordt door het totale aantal ensembles en dat geeft dan de kans op neerslag. (Moltenti, Buizza, Palmer, & Petroliagis, 1994)

5.2.3 Resultaten

Met de opdrachtgever zijn afspraken gemaakt over het bestandsformaat en de opbouw van de resultaten. Een van die afspraken is om een onderscheidt te hebben tussen korte en lange termijn, waarin de korte termijn bestaat uit de eerste 3 dagen na calculatie en de lange termijn bestaat uit dag 4 tot en met 10 na calculatie. Voor de korte termijn wordt een dagelijkse voorspelling getoond terwijl de lange termijn juist een gemiddelde waarde bevat. Om dit mogelijk te maken is er in de modelbuilder per variabelen 2 aanvragen gedaan naar HydroNET waarbij de interval dynamisch wordt ingesteld, op korte termijn is er vast geprogrammeerd dat het interval 1 dag is. Voor de lange termijn is de interval gelijk aan de tijd tussen het start en eindmoment.
Gedurende het onderzoek zijn er 2 versies van de modelbuilder gemaakt, zoals te lezen is in paragraaf 4.5. Het belangrijkste wat dat heeft veroorzaakt is de eenvoud waarmee een gebruiker analyse kan doen met de model builder.
Tijdens de onderzoeksperiode zijn er 2 versies van de modelbuilder gemaakt. De eerste versie was een echte proof of concept, en de tweede versie is veel meer gefocust om de gebruiksvriendelijkheid en foutafhandeling van de code te verbeteren, zie hiervoor Bijlage F. Daarin is duidelijk te zien dat versie 2 veel gebruikersvriendelijker is en het eenvoudiger maakt om aanvragen te doen en het stappenplan van de modelbuilder te doorlopen.

5.2.4 Reactie opdrachtgever

Met de opdrachtgever is gedurende het project meerdere malen contact gezocht om zeker te weten dat het eindproduct naar hun smaak is. Hiertoe zijn er op verscheidene momenten gesprekken gevoerd. De conclusies van deze gesprekken hebben geleid tot twee grote versies. De eerste versie waarbij eisen voldoening het belangrijkst was en een tweede versie waar gebruikersvriendelijkheid centraler stond, lees meer erover in paragraaf 4.5.
Uiteindelijk heeft dit geleid tot een zeer tevreden opdrachtgever die meer kansen en opties ziet met de data dan voor de ontwikkeling van de modelbuilder. Zo kwamen er nieuwe ideeën op als actieve waarschuwingssystemen op voorspellingen.

6 Evaluatie

Om de model builder te verifiëren zijn de cases gebruikt zoals beschreven in Hoofdstuk 5, samen met een review sessie met de opdrachtgever en verificatie van de eisen genoemd in paragraaf 3.5. Uit de sessie na versie 1 zijn we tot de conclusie gekomen dat de eisen zijn voltooid, maar er een verbetering kan zijn op gebruikersvriendelijkheid. Daardoor is er gekozen om nog door te ontwikkelen tot een versie 2. Na versie 2 is er weer een review sessie geweest om de resultaten van de gebruiksvriendelijkheid verbetering te bespreken, welke positief zijn ontvangen.

6.1 Verificatie

Om te zien of de modelbuilder als voltooid beschouwd kan worden is er gekeken naar het programma van eisen en of alle gekozen eisen en wensen naar voren komen in het eindontwerp. Om aan te tonen dat aan al deze eisen is voldaan, is in Bijlage G de lijst zichtbaar met een verwijzing naar het stukje code dat ervoor zorgt dat de eis is gehaald. Aangezien alle eisen en wensen die zijn besloten om meegenomen te worden in het begin gehaald zijn kan de modelbuilder als voltooid worden beschouwd.
Nadat de modelbuilder was voltooid, heeft deze eerste versie “operationeel” gedraaid voor drie weken voordat de klant werd ingelicht. Het voordeel hiervan is dat bugs in deze periode zichtbaar worden die je niet ziet bij incidentele testen. Deze bugs zijn dan ook lastig na te maken maar tonen zich goed op langere termijn en zijn enkel te verwijderen via een proefperiode. Na ongeveer een week kwamen er geen bugs meer naar boven, toch is er voor gekozen om dan nog 2 weken door te testen om zeker te weten dat er niks vreemd meer op gaat komen als die daadwerkelijk gebruikt gaat worden. Nadat dit is bereikt is Weather Impact ingelicht.

6.2 Review sessie

Nadat Weather Impact wist dat de model builder operationeel aan het werk was. Is er een review sessie gepland. In deze sessie is teruggeblikt op het project en of de doelen die er nu leven kloppen met het eindproduct. Dit aangezien wensen van mensen over tijden kunnen veranderen. Daaruit bleek dat ze zeer tevreden waren over het eindproduct echter wel enkele verbeterpuntjes zagen. Zo was in versie 1 het schrijven van een analysescript zeer vatbaar voor typfouten. Een mogelijke oplossing hiervoor is gevonden in klassen programmeren. Dit zorgt ervoor dat de kans op typfouten extreem afneemt. Wat ook naar voren kwam in de 2de korte review sessie. Naast dit punt kwam naar voren dat Weather Impact vooral zeer enthousiast was over het behaalde resultaat in de modelbuilder en de opties die deze biedt.

6.3 Controle werking

Aangezien halverwege mei de model builder operationeel draaide is er nog een proefperiode achteraan geweest. In deze proefperiode zijn er opties getest en de cases geprobeerd beschreven in hoofdstuk 5. Door deze cases zijn nog een aantal kleine bugs en waarschuwing aan het ligt gekomen. De grootste hiervan ontstaan als HydroNET gewijzigd wordt. Aangezien de modelbuilder gekoppeld zit aan HydroNET 4 bèta, welke ook nog actief wordt ontwikkeld, willen er nog weleens grote interne wijzigingen plaatsvinden. Deze kunnen ervoor zorgen dat de modelbuilder vastloopt of crasht. Momenteel is hier nog geen automatische oplossing voor en dient er contact gezocht te worden met de ontwikkeling van HydroNET om het probleem op te lossen. In de toekomst zijn er opties dat je een time-out in bouwt in de modelbuilder waardoor deze zichzelf herstart, mocht er dan alsnog een probleem aan de HydroNET kant zijn, is de ontwikkelingsafdeling alsnog noodzakelijk. Een andere optie is om de modelbuilder door te zetten naar de productie versie van HydroNET. De productie versie van HydroNET is een stabiele versie die gereed is voor commercieel gebruik. Hierdoor is de kans op problemen kleiner en worden deze ook altijd al opgelost.

7 Aanbevelingen

Door de beperkte tijd is het resultaat van het onderzoek nog steeds enkel een prototype. Het gevolg hiervan is dan ook dat deze oplossing mogelijk niet perfect werkt of de beste oplossing is. Het geeft wel heel ruim de voordelen van zo'n systeem aan en de kracht voor analyses. Echter zijn er nogal een aantal punten waar verbetering gewenst is om het tot een product te maken.

7.1 Veiligheid

Allereerst is er verbetering gewenst op het gebied van beveiliging. In dit geval vooral inloggegevens. Op het moment staat er binnen de modelbuilder voor het uploaden naar een ftp en het versturen van mails allerlei inloggegevens in het script. Hierbij is de overweging gemaakt om dit op te slaan onder encryptie of in de wachtwoorden niet direct zichtbaar te maken. Echter is dit een heleboel schijnveiligheid als je dat toepast door de openheid van Python. Dit komt omdat de inlog het wachtwoord in een string vereist, dus als je het wachtwoord geëncrypt in de code zet. Dien je die te decrypten voor de inlog. Hierdoor kun je natuurlijk gewoon dat stukje code uitvoeren en heeft de encryptie geen zin. Vanwege deze redenen is ervoor gekozen om het gewoon als strings op te slaan ondanks alle bezwaren in verband met de veiligheid. Dit wordt enigszins opgevangen doordat de scripts enkel intern beschikbaar zijn. Maar dit mag in de toekomst netter gemaakt worden voor een product.

7.2 Foutafhandeling

Ten tweede is het van belang om nog beter te kijken naar de foutafhandeling. Op het moment worden voor vrij veel fouten mails verstuurd. Echter is het een goed idee om een programmeur eens naar de code te laten kijken en deze te laten controleren of daadwerkelijk alle mogelijke fouten worden afgevangen. Als dat niet het geval is, is het verstandig hier meer aan te doen, zodat het voor de gebruiker duidelijk is waarom het programma niet werkt.

7.3 Data aanvraag limiteren

Ten derde is er ruimte voor verbetering in controle van de geschatte data grootte. Op het moment zou je plotseling zoveel data kunnen aanvragen dat de server daarvan crasht. Om dit op te vangen kan er in de toekomst voornamelijk aan de server zijde een controle komen die dit opvangt. Momenteel is het beste om het prototype alleen te gebruiken voor die mensen die begrijpen wat ze opvragen en wanneer om onnodige serverlast te voorkomen.

8 Conclusie

Er zijn vele verschillende wensen voor de model builder voor HydroNET. Daarbij hebben de betrokken partijen (Bedrijfsvoering, Ontwikkeling, Hydrologen en Weather Impact) dezelfde visie voor de lange termijn, maar op korte termijn verschillen de meningen flink. Uit deze lijst met wensen zijn er een aantal gekozen die op korte termijn het belangrijkst zijn om daarmee een “proof of concept” te creëren.
Het gecreëerde model builder prototype is duidelijk in staat om aan alle wensen te voldoen die uitgekozen zijn. Het belangrijkste aan deze model builder is dat duidelijk alle potentie van ’een dergelijk product getoond kan worden.
De belangrijkste verbeteringsslagen voor de Hydro Informatica dat de model builder te weeg brengt zijn: het maken van een experimenteeromgeving, analyseren van meerdere datasoorten en het uit kunnen voeren van analyses die op korte termijn grote veranderingen ondergaan.
Het idee van de experimenteeromgeving is dat het hydrologen in staat stelt om bij ideeën voor potentiele applicaties zelf alvast te onderzoeken of het zinnig is om de applicatie te ontwikkelen. Daarnaast ook hoe deze het beste kan functioneren en welk soort probleem deze applicatie gaat ondersteunen of voor welke oplossingen deze kan gaan zorgen.
Concluderend, de model builder is een grote toevoeging voor HydroNET en zal dus ook het beste verder ontwikkeld kunnen worden. Daarbij is het van belang voor de korte termijn vooral tijd te investeren aan de genoemde aanbevelingen in hoofdstuk 7. Daarnaast blijkt uit het prototype dat de toevoeging van een model builder groter zijn dan gedacht en dat het zeker zinnig is om tijd in te steken.
Uiteindelijk heeft de model builder ervoor gezorgd dat de methode van het ophalen en verwerken van data uit HydroNET mogelijk is op een flexibele en gestandaardiseerde methode. Deze standaardisatie zorgt voor eenvoud en gebruiksvriendelijkheid waardoor experimenteren met ideeën voor data analyse stukken makkelijker is.

9 Bibliografie

Anderson, C., & Drossopoulou, S. (2004, September 28). BabJ: From Object Based to Class Based Programming via Types. 53-81. Warsaw, Polen.

Broom, J., English, J., Marois, J., Latorre, B., & Aviles, J. (1995). Development of an infection Model for Botrytis Bunch Rot fo Grapes Based on Wetness Duration and Temperature. *Phytopathology 85*, 97-102.

Brouwer, H. (2000). Integrated water management: emerging issues and challenges. *Agricultural Water Management*, 217-228.

Fielding, R., Gettys, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). *Hypertext Transfer Protocol -- HTTP/1.1.* The Internet Society.

HydroNET BV. (2015). *HydroNET 4 server.* Amersfoort: Hydrologic BV.

Lobbrecht, A., Einfalt, T., Reichard, L., & Poortinga, I. (2011). Decision support for urban drainage using radar data of HydroNET-Scout. *Weather Radar and Hydrology* (pp. 1-6). Exeter: IAHS.

Moltenti, F., Buizza, R., Palmer, T., & Petroliagis, T. (1994). The ECMWF Ensemble Predicition System: Methodology and validation. *European Centre for Medium-Range Weather Forecasts, UK*, 73-119.

Van Der Burgt, F., & Van Pelt, S. (2016, April 4). Discovery of the assignment. (Y. Fredrix, Interviewer)

Van Der Wielen, J. (2015). *HydroNET 4 Server API and WMS calls.* Amersfoort: HydroLogic Systems.

Handige links voor ontwikkeling

http://www.tutorialspoint.com/python/index.htm
https://docs.python.org/3/

Bijlage A Programma van Eisen

Tabel 2: Het totale programma van eisen

Wat	Eis/Wens	Wie?	Belangrijk
Model builder gebruikt python scripts	Eis	WI	Belangrijkste
Model builder heeft goed documentatie	Eis	Iedereen	Belangrijkste
Model builder kan zelfstandig ECMWF data inlezen en gebruiken	Eis	WI	Belangrijkste
Hydrologen in staat stellen om met een lage drempel experimenten te doen vanuit alle databronnen	Eis	Bedrijfsvisie	Belangrijkste
Model builder is in staat gegevens op te halen	Eis	Business	Belangrijkste
Modelbuilder is een UI op bestaande HydroNET	Eis	Ontwikkeling	Belangrijkste
Modelbuilder laat dataflows creëren	Eis	Ontwikkeling	Belangrijkste
Modelbuilder maakt HydroNET beheersbaar	Eis	Ontwikkeling	Belangrijkste
Werkt met alle data soorten	Wens	Iedereen	Belangrijkste
Model Builder kan automatisch op dagelijkse interval worden uitgevoerd	Eis	WI	Belangrijkste
Analyse moet live draaien of periodiek	Wens	Business	Belangrijk
Analyses voor Water control room	Wens	Business	Belangrijk
Complexere analyses mogelijk maken	Eis	Bedrijfsvisie	Belangrijk
De model builder moet de uitkomsten plaatsen in HydroNET	Wens	Iedereen	Belangrijk
Model builder draait scripts op scheduled moment	Wens	Business	Belangrijk
Model builder functioneert buiten HydroNET (ook bronnen buiten HydroNET)	Wens	Potentiële gebruiker	Belangrijk
Model builder is bruikbaar intern en partners	Wens	Bedrijfsvisie	Belangrijk
Model builder is in staat gegevens te bewerken	Eis	Business	Belangrijk
Model builder is onderdeel van HydroNET	Eis	Ontwikkeling	Belangrijk
Modelbuilder faciliteert de ontlasting van ontwikkelaars voor uitgebreide analyses	Eis	Potentiële gebruiker	Belangrijk
Modelbuilder maakt analyses beheersbaar	Eis	Potentiële gebruiker	Belangrijk

Moet decision support geven	Eis	Business	Belangrijk
Resultaten kunnen in HydroNET	Eis	Bedrijfsvisie	Belangrijk
Werkt met meerder scripting language	Wens	Business	Belangrijk
Model builder kan ook andere data inlezen op keuze van de gebruiker	Wens	WI	Zeer gewenst
Model builder moet de uitkomsten op een ftp server kwijt kunnen	Wens	WI	Zeer gewenst
Model builder draait op HydroNET server	Wens	WI	Wenselijk
Model builder kan R gebruiken	Wens	Business	Wenselijk
Kan grafisch programmeerbaar zijn	Wens	Business	Wenselijk
Klant kan scripts maken en uploaden naar HydroNET	Wens	Business	Wenselijk
Model builder is in staat gegevens in HydroNET te zetten	Wens	Business	Wenselijk
Model builder ondersteund meerdere scripts	Wens	Potentiële gebruiker	Wenselijk
Operationeel script in HydroNET	Wens	Potentiële gebruiker	Wenselijk
ÙI tools, soort van wizard	Wens	Business	Wenselijk
Gebaseerd op bestaande ideeën	Optie	Business	

In Tabel 2 zijn alle eisen op een rijtje gezet en gerangschikt op belang. De meest belangrijke eisen staan bovenaan en de minst belangrijke onderaan. Hieruit is een selectie gemaakt welke eisen binnen de scope van het prototype vallen en welke niet, deze zijn terug te vinden in hoofdstuk 3.5. Daarbij is ook een uitleg beschikbaar waarom bepaalde eisen wel en niet meegenomen worden.

Bijlage B Conceptuele data flow

Binnen het onderzoek zijn een aantal conceptuele data flows bedacht om zo de beste oplossing te bepalen. De dataflows die het meest identiek zijn, zijn zichtbaar in de komende paragrafen. Op deze dataflows zijn vele kleine aanpassingen/extra's te bedenken, zoals: het upload deel naar HydroNET kan natuurlijk ook naar enige andere server met kleine aanpassingen in de broncode.

B.1 Conceptuele model 1

Zoals zichtbaar in Fig. 12 is dit een dataflow voor het eerste conceptuele model. De dataflow bestaat uit verschillende blokken; zo is het rondje de start van de data flow, een vierkant is een interface, en een ellips is een proces dat verder zonder gebruikersopties zich uitvoert. In deze versie komt het er dus op neer dat er via de API data wordt gedownload, deze data arriveert in JSON en wordt dan voor python gereed gemaakt waarna analyses plaats vinden. Deze analyse resultaten worden dan weer in JSON gezet om weer naar HydroNET te uploaden.

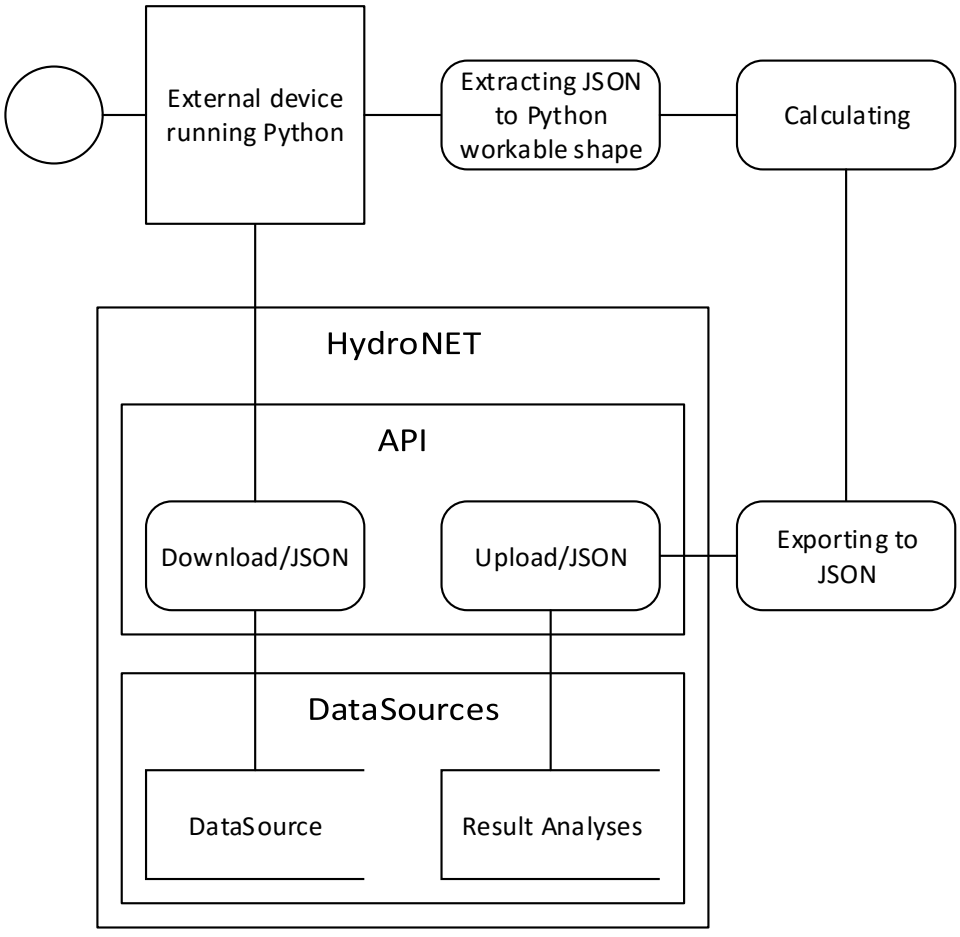

Fig. 12: Conceptuele dataflow 1; rondje is de start; waarna data meegaat vanaf download naar de upload

B.2 Conceptuele model 2

In Fig. 13 is zichtbaar een 2de dataflow idee. Dit idee is gebaseerd op de processen binnen HydroNET en heeft ook als doel om het geheel binnen HydroNET te houden. Dit gebeurt door verschillende processoren binnen HydroNET aan te roepen. (Die nog niet bestaan) Door deze processoren is het mogelijk om data te selecteren met python en deze dan op te halen uit de datasources. Iedere datasource geeft zo intern zijn data af aan het script waarna deze geanalyseerd kunnen worden in python. De resultaten van deze analyse worden dan gekopieerd naar een nieuwe databron vanaf waar deze dan zichtbaar wordt in de verschillende apps. En te gebruiken is voor de download API.
Het nadeel aan deze methode is dat je veel kennis moet hebben van .NET systemen om het op te zetten. Daarnaast dat scripts mogelijk de kwaliteit en stabiliteit van HydroNET in gevaar kunnen brengen. Dit zijn opties die je niet wilt, hierdoor zal zo'n oplossing dus gemaakt moeten worden door een ervaren programmeur.

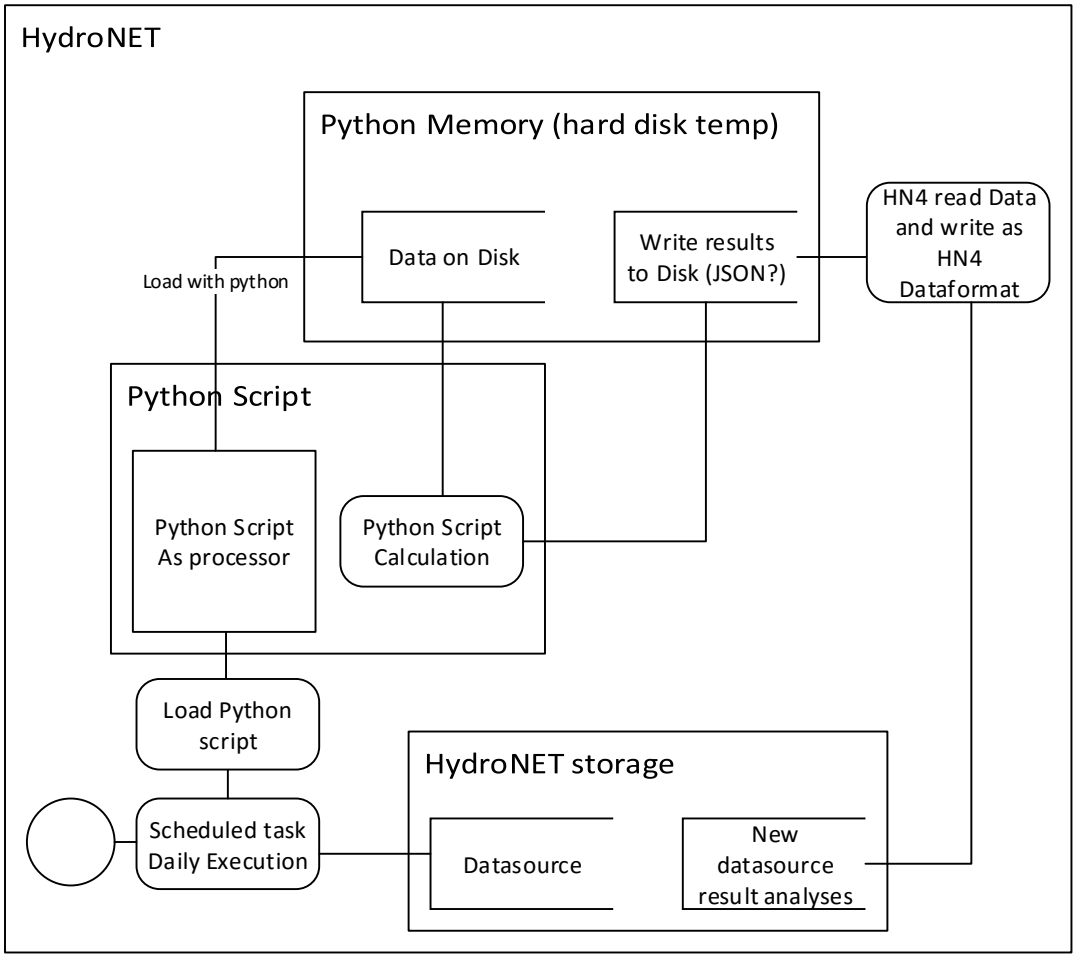

Fig. 13:Conceptuele dataflow 2: Rondje geeft de start waarna je data flow volgt vanaf load richting write

Bijlage C Model Builder uitwerking

In deze bijlage zijn gedeeltes code aanwezig met uitleg. Het belang hiervan is dat er een begrijpelijke uitleg ligt over de implementatie van de modelbuilder in Python. Hierin geeft paars een functie weer, groen comments, rood stukken ingeklapte code en blauw code.

C.1 Versie 1

```
def JSONWIModelGrid(Data,StartDates,EndDates,GridDefinition,MyVar,Interval,Unit):
    StartVector=[]
    EndVector=[]
    for j in range(len(StartDates)):
        StartVector.append(StartDates[j].isoformat())
        EndVector.append(EndDates[j].isoformat())
    x=len(Data[0,:,0])
    y=len(Data[0,0,:])
    JSON={"Data":{},"GridDefinition":GridDefinition,"MetaData":{'Variable-
Name':MyVar,'StartDate':StartVector[0],'EndDate':EndVector[-1],'Interval':Inter-
val,'Unit':Unit}}
    for i in range(len(Data[:,0,0])):
        JSON["Data"][str(i)]={}
        JSON["Data"][str(i)]["StartDate"]=StartVector[i]
        JSON["Data"][str(i)]["EndDate"]=EndVector[i]
        A=np.reshape(Data[i,:,:],(1,x*y)).tolist()
        JSON["Data"][str(i)]["Data"]=A[0]
    return JSON

def dateFormat(Date):
    if Date.month<10:
        MonthStr="0"+str(Date.month)
    else:
        MonthStr=str(Date.month)
    if Date.day<10:
        DayStr="0"+str(Date.day)
    else:
        DayStr=str(Date.day)
    if Date.hour<10:
        HourStr="0"+str(Date.hour)
    else:
        HourStr=str(Date.hour)
    if Date.minute<10:
        MinuteStr="0"+str(Date.minute)
    else:
        MinuteStr=str(Date.minute)
    DateStr=str(Date.year)+MonthStr+DayStr+HourStr+MinuteStr+"00"
```

```
    return DateStr
def returnGridData(MyType,MyDatasourcename,MyDate,MyVarcode,Interval,receiv-
ers,locatie='Kantoor',Extent="NotAvailable",Processor="NotAvailable",Calculation-
Type="NotAvailable"):
    '''This functions allows for GridData requests to the HydroNET 4 server.
    It uses 6 obligated arguments and 4 free argument.
    The First argument is the location you are currently at, the office or somewhere else.
    The First argument is the datasourcetype.
    The Second argument is the datasourcename.
    The Third argument is the date in a list starting at ModelDate or StartDate and followed
by the necessary date
    The Fourth argument is a list of all the variablescodes requested
    The Fifth argument is a dict containing the interval type and value
    The Sixth argument is a list of the receivers for mail adresses
    The rest are free arguments, they can be in any order as long as you start with Thing=
    A location can be added in the form locatie=, at the office network this doesn't have to be
use
    An extent can be added as a correct dictionary in the form Extent={}
    A processor to edit the data from HydroNET can be added as well in a dictionary stating
the processor data, in the following form Processor={}
    And a calculationType can be added, this calculationtype can deliver stuff as minima
and maxima; created by adding CalculationType='' '''
Maken requests voor HydroNET
    r = requests.post(url,headers=headers,data=json.dumps(payload))
    if r.status_code!=requests.codes.ok:
        errormail(receivers,'API Request Failed',message)
    data=r.json()
    #Metadata
Veranderen Response in Python formaat voor de output
    return timevec, EndVec, GridDefinition, latvec, lonvec, forecast, description, interval, is-
cum, Units,data
```

C.2 Versie 2

```
class GridData:                    # Class in which all kind of Grid Data can be stored
  ''' The class has several parts consisting:
```

- StartTimes
- EndTimes
- GridDefinitions
- Latitudes
- Longitudes
- Data
- Description of variables
- Interval
- Is the data cumalitive?
- The units of the variables
- The results from the HydroNET request
- The filenames
- Results for upload can be added. Next to this has the class got a method, which makes the JSON format for Weather Impact'''

```
  def __init__(self,StartTimes, EndTimes, GridDefinition, latvec, lonvec, dataPython, description, interval, iscum, Units,Dictionary,FileName):
    self.StartTimes=StartTimes
    self.EndTimes=EndTimes
    self.Grid=GridDefinition
    self.Latitude=latvec
    self.Longitude=lonvec
    self.Data=dataPython
    self.Description=description
    self.Interval=interval
    self.Cumalitive=iscum
    self.Units=Units
    self.HydroNETFormat=Dictionary
    self.Name=FileName
    self.Results=[]

  def JSONWI(self,NewData=[[-9999,-9999]]):
    ALLJSON=[]
    StartDates=self.StartTimes
    EndDates=self.EndTimes
    StartVector=[]
    EndVector=[]
    for l in range(len(NewData)):
      for j in range(len(StartDates)):
        StartVector.append(StartDates[j].isoformat())
        EndVector.append(EndDates[j].isoformat())
      for k in range(len(self.Description)):
        JSON=[]
        if type(NewData[0]) is list:
          Data=self.Data[k]
        else:
          Data=NewData[l]
```

```
            x=len(self.Longitude)
            y=len(self.Latitude)
            JSON={"Data":{},"GridDefinition":self.Grid[k],"MetaData":{'Variable-
Name':self.Description[k],'StartDate':StartVector[0],'EndDate':EndVector[-1],'Inter-
val':self.Interval[k],'Unit':self.Units[k]}}
            for i in range(len(Data[:,0,0])):
               JSON["Data"][str(i)]={}
               JSON["Data"][str(i)]["StartDate"]=StartVector[i]
               JSON["Data"][str(i)]["EndDate"]=EndVector[i]
               A=np.reshape(Data[i,:,:],(1,x*y)).tolist()
               JSON["Data"][str(i)]["Data"]=A[0]
            ALLJSON.append(JSON)
         self.Results=ALLJSON
      return

def dateFormat(Date):
   '''This function changes the date format from a datetime format to a string
   This string is being made in the format for HydroNET
   This follows a year out of 4 characters
         A month of 2 characters
         A day of 2 characters
         A hour of 2 characters
         Minutes of 2 characters
         Seconds of 2 characters
   The reaction is then a string consisting of the elements above after each other. '''
   if Date.month<10:
      MonthStr="0"+str(Date.month)
   else:
      MonthStr=str(Date.month)
   if Date.day<10:
      DayStr="0"+str(Date.day)
   else:
      DayStr=str(Date.day)
   if Date.hour<10:
      HourStr="0"+str(Date.hour)
   else:
      HourStr=str(Date.hour)
   if Date.minute<10:
      MinuteStr="0"+str(Date.minute)
   else:
      MinuteStr=str(Date.minute)
   DateStr=str(Date.year)+MonthStr+DayStr+HourStr+MinuteStr+"00"
   return DateStr
```

```
def returnGridData( MyType,MyDatasourcename,MyDate,MyVarcode,Interval,receiv-
ers,locatie='Kantoor',Extent="NotAvailable",Processor="NotAvailable",Calculation-
Type="NotAvailable",Name=""):
   '''This functions allows for GridData requests to the HydroNET 4 server.
   It uses 6 obligated arguments and 5 free argument.
   The First argument is the datasourcetype.
   The Second argument is the datasourcename.
   The Third argument is the date in a list starting at ModelDate or StartDate and followed
by the necessary date
   The Fourth argument is a list of all the variablescodes requested
   The Fifth argument is a dict containing the interval type and value
   The Sixth argument is a list of the receivers for mail adresses
   The rest are free arguments, they can be in any order as long as you start with Thing=
   A location can be added in the form locatie=, at the office network this doesn't have to be
use
   An extent can be added as a correct dictionary in the form Extent={}
   A processor to edit the data from HydroNET can be added as well in a dictionary stating
the processor data, in the following form Processor={}
   A calculationType can be added, this calculationtype can deliver stuff as minima and
maxima; created by adding CalculationType="
         And last the Name of the files for sending to the server can be given as
Name='TemperatureMin.JSON'  '''
Maken request voor HydroNET
   if r.status_code!=requests.codes.ok:
      mail.errormail(receivers,'API Request Failed',message)
      sys.exit("Wrong data request, API failure")
   print('Deriving the variables')
   data=r.json()
Response in Python formaat zetten voor de parameters in de class
   DataSet=GridData(timevec, EndVec, GridDefinition, latvec, lonvec, forecast, description,
interval, iscum, Units ,data ,Name)
   return DataSet
```

Bijlage D JSON formaat


```
{
	"Data": {
		"1": {
			"Data": [DataDag2],
			"StartDate": "2016-06-16T12:00:00",
			"EndDate": "2016-06-17T12:00:00"
		},
		"0": {
			"Data": [DataDag1],
			"StartDate": "2016-06-15T12:00:00",
			"EndDate": "2016-06-16T12:00:00"
		},
		"2": {
			"Data": [DataDag3],
			"StartDate": "2016-06-17T12:00:00",
			"EndDate": "2016-06-18T12:00:00"
		}
	},
	"MetaData": {
		"Unit": "MM",
		"Interval": {
			"Value": 1.0,
			"Type": "Days"
		},
		"VariableName": "Precipitation",
		"StartDate": "2016-06-15T12:00:00",
		"EndDate": "2016-06-18T12:00:00"
	},
	"GridDefinition": {
		"Rows": 49,
		"Name": "Ethiopia Ecmwf Eps 6h",
		"EndColumn": 64,
		"Yur": 15.125,
		"CellHeight": 0.25,
		"GridDefinitionId": 20,
		"EndRow": 48,
		"Xll": 32.375,
		"StartColumn": 0,
		"Xur": 48.625,
		"StartRow": 0,
		"ProjectionId": 3,
		"CellWidth": 0.25,
		"Yll": 2.875,
		"Columns": 65
	}
}
```

Bijlage E Grape Compass Case

Zoals beschreven in paragraaf 5.1 is er gekozen om de case van het Grape Compass gedeeltelijk uit te werken. In het volgende deel wordt zeer goed duidelijk wat het gevolg is van de iteratieslag. De stukjes code zijn kleur gecodeerd; rood is download, blauw is analyse, groen is formaat maken en opslaan. Deze case is uitgewerkt in v1 van de Model Builder.

E.1 Uitwerking Grape Compass

```
StartModeling=datetime.datetime(year=2016,month=2,day=20,hour=12,minute=0,second=0)
StartStr=MB.dateFormat(StartModeling)
ModelingDate=StartModeling
Spath="GrapeCompassBest\\"
if not os.path.exists(Spath):
   os.makedirs(Spath)
os.chdir(Spath)
LWDpath="LWD"+'\\'
RHpath="RH"+"\\"
IIpath="Botrytis"+"\\"
while ModelingDate<datetime.datetime(year=2016,month=2,day=21,hour=12,minute=0,second=0):
   ModeldateStr=MB.dateFormat(ModelingDate)
   [timevecECMWF, EndVecECMWF,GridECMWF, latvecECMWF, lonvecECMWF, dataECMWF, descriptionTEMP, intervalid, iscum, Units, JSONECMWF]=MB.returnGridData("EnsembleGrid","ECMWF.EPS.STellenbosch",["ModelDate",ModeldateStr],["DPT","TMP"],["Hours","3"],"mail")
   dataDPT=dataECMWF[0,:,:,:,:]
   dataTMP=dataECMWF[1,:,:,:,:]
   #Initialisatie
   ea=np.zeros((len(dataDPT[:,0,0,0]),len(timevecECMWF)+1,len(latvecECMWF),len(lonvecECMWF)))
   e0=np.zeros((len(dataDPT[:,0,0,0]),len(timevecECMWF)+1,len(latvecECMWF),len(lonvecECMWF)))
   RHe=np.zeros((len(dataDPT[:,0,0,0]),len(timevecECMWF)+1,len(latvecECMWF),len(lonvecECMWF)))
   RH=np.zeros((len(timevecECMWF)+1,len(latvecECMWF),len(lonvecECMWF)))
   LWD=np.zeros((len(timevecECMWF)+1,len(latvecECMWF),len(lonvecECMWF)))
   dataTMP90=np.zeros((len(timevecECMWF)+1,len(latvecECMWF),len(lonvecECMWF)))
   TMP=np.zeros((len(timevecECMWF)+1,len(latvecECMWF),len(lonvecECMWF)))
   II=np.zeros((len(timevecECMWF)+1,len(latvecECMWF),len(lonvecECMWF)))
   #Controle of berekeningsaannames correct zijn
   if intervalid[0]["Type"]=="Hours":
      if intervalid[0]["Value"]<3:
         print("This does not work")
```

```
    else:
        print("This does not work")
    # Inladen vorige tijdstap bij nieuwe dag
    #    if ModeldateStr!=StartStr:
    Previous=MB.dateFormat(ModelingDate+datetime.timedelta(days=0, seconds=0, micro-
seconds=0, milliseconds=0, minutes=0, hours=-intervalid[0]["Value"]))
    os.chdir(LWDpath)
    LWD[0,:,:]=np.genfromtxt(Previous+".csv",delimiter=',')
    basepath=os.path.dirname(os.getcwd())
    os.chdir(basepath)
    # Berekening van RH en wanneer RH>90 de tijd optellen
    for l in range(len(dataDPT[:,0,0,0])):
        for i in range(1,len(timevecECMWF)+1):
            for j in range(len(latvecECMWF)):
                for k in range(len(lonvecECMWF)):
                    ea[l,i,j,k]=0.6108*math.exp((17.27*dataDPT[l,i-1,j,k])/(dataDPT[l,i-1,j,k]+237.3))
                    e0[l,i,j,k]=0.6108*math.exp((17.27*dataTMP[l,i-1,j,k])/(dataTMP[l,i-1,j,k]+237.3))
                    RHe[l,i,j,k]=100*ea[l,i,j,k]/e0[l,i,j,k]
    for i in range(1,len(timevecECMWF)+1):
        for j in range(len(latvecECMWF)):
            for k in range(len(lonvecECMWF)):
                RH[i,j,k]=sorted(RHe[:,i,j,k])[45]
                dataTMP90[i-1,j,k]=sorted(dataTMP[:,i-1,j,k])[45]
                if RH[i,j,k]>90:
                    LWD[i,j,k]=LWD[i-1,j,k]+intervalid[0]["Value"]
                else:
                    LWD[i,j,k]=0
                if dataTMP90[i-1,j,k]<12:
                    TMP[i,j,k]=12
                elif dataTMP90[i-1,j,k]>32 and dataTMP90[i-1,j,k]<40:
                    TMP[i,j,k]=32
                elif dataTMP90[i-1,j,k]>40:
                    TMP[i,j,k]=40
                else:
                    TMP[i,j,k]=dataTMP90[i-1,j,k]
                II[i,j,k]=-2.647866-0.374927*LWD[i,j,k]+0.061601*LWD[i,j,k]*TMP[i,j,k]-
0.001511*LWD[i,j,k]*math.pow(TMP[i,j,k],2)
                if LWD[i,j,k]>16:
                    II[i,j,k]=1
        if i!=1:
            if not os.path.exists(LWDpath):
                os.makedirs(LWDpath)
            os.chdir(LWDpath)
            np.savetxt(str(MB.dateFormat(timevecECMWF[i-1]))+".csv",LWD[i],delimiter=',')
            basepath=os.path.dirname(os.getcwd())
            os.chdir(basepath)
            if not os.path.exists(RHpath):
```

```
        os.makedirs(RHpath)
    os.chdir(RHpath)
    np.savetxt(str(MB.dateFormat(timevecECMWF[i-1]))+".csv",RH[i],delimiter=',')
    basepath=os.path.dirname(os.getcwd())
    os.chdir(basepath)
    if not os.path.exists(IIpath):
        os.makedirs(IIpath)
    os.chdir(IIpath)
    np.savetxt(str(MB.dateFormat(timevecECMWF[i-1]))+".csv",II[i],delimiter=',')
    basepath=os.path.dirname(os.getcwd())
    os.chdir(basepath)
ModelingDate=ModelingDate+datetime.timedelta(1)
```

Bijlage F ECMWF Ethiopië case

Zoals beschreven in hoofdstuk 5.2 is er gekozen om de case van de ECMWF waardes volledig uitwerkingen. In de volgende 2 delen wordt zeer goed duidelijk wat het gevolg is van de iteratieslag. Vooral de gebruiksvriendelijkheid is verbeterd, zoals zichtbaar is in de hoeveelheid code. De stukjes code zijn kleur gecodeerd; rood is download, blauw is analyse, groen is formaat maken en uploaden. De belangrijkste verschillen zijn vetgedrukt.

F.1 Uitwerking in Modelbuilder v1

```
import datetime
import numpy as np
import ModelBuilder as MB

Today=datetime.datetime.now()
ModelDay=Today+datetime.timedelta(-1)
ModelDay=ModelDay.replace(hour=12,minute=0)
StartDate1=ModelDay+datetime.timedelta(1)
StartDate2=ModelDay+datetime.timedelta(4)
EndDate1=ModelDay+datetime.timedelta(4)
EndDate2=ModelDay+datetime.timedelta(10)

ModelDayStr=MB.dateFormat(ModelDay)
StartDate1Str=MB.dateFormat(StartDate1)
StartDate2Str=MB.dateFormat(StartDate2)
EndDate1Str=MB.dateFormat(EndDate1)
EndDate2Str=MB.dateFormat(EndDate2)

[StartTmax1, EndTmax1, GridDefinitionTmax1, latvecTmax, lonvecTmax, forecastTmax,
descriptionTmax, intervalTmax, iscumTmax, UnitsTmax,
JSONDataTmax]=MB.returnGridData("ModelGrid","CommonSense.Ethiopia.Ecmwf.Deterministic.6h",["ModelDate+StartDate",ModelDayStr,StartDate1Str,EndDate1Str],["TMAX","InstantaneousWindGust"],["Days","1"],"errormail",CalculationType="Max")
[StartTmax2, EndTmax2, GridDefinitionTmax2, latvecTmax2, lonvecTmax2,
forecastTmax2, descriptionTmax2, intervalTmax2, iscumTmax2,UnitsTmax2,
JSONDataTmax2]=MB.returnGridData("ModelGrid","CommonSense.Ethiopia.Ecmwf.Deterministic.6h",["ModelDate+StartDate",ModelDayStr,StartDate2Str,EndDate2Str],["TMAX",
"InstantaneousWindGust"],["Days",str((EndDate2-
StartDate2).days)],"errormail",CalculationType="Max")
[StartTmin1, EndTmin1, GridDefinitionTmin1, latvecTmin, lonvecTmin, forecastTmin,
descriptionTmin, intervalTmin, iscumTmin,UnitsTmin,
JSONDataTmin]=MB.returnGridData("ModelGrid","CommonSense.Ethiopia.Ecmwf.Deterministic.6h",["ModelDate+StartDate",ModelDayStr,StartDate1Str,EndDate1Str],["TMIN"],["Days","1"],"errormail",CalculationType="MIN")
```

[StartTmin2, EndTmin2, GridDefinitionTmin2, latvecTmin2, lonvecTmin2, forecastTmin2, descriptionTmin2, intervalTmin2, iscumTmin2,UnitsTmin2, JSONDataTmin2]=MB.returnGridData("ModelGrid","CommonSense.Ethiopia.Ecmwf.Deterministic.6h",["ModelDate+StartDate",ModelDayStr,StartDate2Str,EndDate2Str],["TMIN"],["Days",str((EndDate2-StartDate2).days)],"errormail",CalculationType="MIN")
[StartP1, EndP1, GridDefinitionP1, latvecP1,lonvecP1,forecastP1,descriptionP1,intervalP1, iscumP1,UnitsP1,JSONADataP1]=MB.returnGridData("EnsembleGrid","CommonSense.Ethiopia.Ecmwf.Eps.6h",["ModelDate+StartDate",ModelDayStr,StartDate1Str,EndDate1Str],["P"],["Days","1"],"errormail")
[StartP2, EndP2, GridDefinitionP2, latvecP2,lonvecP2,forecastP2,descriptionP2,intervalP2, iscumP2,UnitsP2,JSONADataP2]=MB.returnGridData("EnsembleGrid","CommonSense.Ethiopia.Ecmwf.Eps.6h",["ModelDate+StartDate",ModelDayStr,StartDate2Str,EndDate2Str],["P"],["Days",str((EndDate2-StartDate2).days)],"errormail")

```
Pminpercent=30/100.
minind=int(np.floor((np.shape(forecastP1[0,1:50,:,:,:])[0])*Pminpercent))
forecast_pmin1=np.sort(forecastP1[0,1:50,:,:,:],0)[minind]
minind=int(np.floor((np.shape(forecastP2[0,1:50,:,:,:])[0])*Pminpercent))
forecast_pmin2=np.sort(forecastP2[0,1:50,:,:,:],0)[minind]

Pmaxpercent=90/100.
maxind=int(np.floor((np.shape(forecastP1[0,1:50,:,:,:])[0])*Pmaxpercent))
forecast_pmax1=np.sort(forecastP1[0,1:50,:,:,:],0)[maxind]
maxind=int(np.floor((np.shape(forecastP2[0,1:50,:,:,:])[0])*Pmaxpercent))
forecast_pmax2=np.sort(forecastP2[0,1:50,:,:,:],0)[maxind]

pthres=0.5
forecast_plikely1=np.zeros(np.shape(forecastP1[0,1:50,:,:,:])[1:])

for i in range(len(forecastP1[0,0,:,0,0])):
   for la in range(len(forecastP1[0,0,0,:,0])):
      for lo in range(len(forecastP1[0,0,0,0,:])):

forecast_plikely1[i,la,lo]=len(forecastP1[0,1:50,i,la,lo][forecastP1[0,1:50,i,la,lo]>=pthres])/float(len(forecastP1[0,1:50,i,la,lo]))*100

forecast_plikely2=np.zeros(np.shape(forecastP2[0,1:50,:,:,:])[1:])

for la in range(len(forecastP2[0,0,0,:,0])):
   for lo in range(len(forecastP2[0,0,0,0,:])):

forecast_plikely2[0,la,lo]=len(forecastP2[0,1:50,0,la,lo][forecastP2[0,1:50,0,la,lo]>=pthres])/float(len(forecastP2[0,1:50,0,la,lo]))*100
```

JSONTmax=MB.JSONWIModelGrid(forecastTmax[0],StartTmax1,EndTmax1,GridDefinitionTmax1[0],descriptionTmax[0],intervalTmax[0],UnitsTmax[0])
JSONTmaxLong=MB.JSONWIModelGrid(forecastTmax2[0],StartTmax2,EndTmax2,GridDefinitionTmax2[0],descriptionTmax2[0],intervalTmax2[0],UnitsTmax2[0])
JSONTmin=MB.JSONWIModelGrid(forecastTmin[0],StartTmin1,EndTmin1,GridDefinitionTmin1,descriptionTmin,intervalTmin[0],UnitsTmin[0])
JSONTminLong=MB.JSONWIModelGrid(forecastTmin2[0],StartTmin2,EndTmin2,GridDefinitionTmin2,descriptionTmin2,intervalTmin2[0],UnitsTmin2[0])
JSONPmin=MB.JSONWIModelGrid(forecast_pmin1,StartP1,EndP1,GridDefinitionP1,descriptionP1,intervalP1[0],UnitsP1[0])
JSONPmax=MB.JSONWIModelGrid(forecast_pmax1,StartP1,EndP1,GridDefinitionP1,descriptionP1,intervalP1[0],UnitsP1[0])
JSONPlikely=MB.JSONWIModelGrid(forecast_plikely1,StartP1,EndP1,GridDefinitionP1,descriptionP1,intervalP1[0],UnitsP1[0])
JSONPminLong=MB.JSONWIModelGrid(forecast_pmin2,StartP2,EndP2,GridDefinitionP2,descriptionP2,intervalP2[0],UnitsP2[0])
JSONPmaxLong=MB.JSONWIModelGrid(forecast_pmax2,StartP2,EndP2,GridDefinitionP2,descriptionP2,intervalP2[0],UnitsP2[0])
JSONPlikelyLong=MB.JSONWIModelGrid(forecast_plikely2,StartP2,EndP2,GridDefinitionP2,descriptionP2,intervalP2[0],UnitsP2[0])
JSONWindmax=MB.JSONWIModelGrid(forecastTmax[1],StartTmax1,EndTmax1,GridDefinitionTmax1[1],descriptionTmax[1],intervalTmax[0],UnitsTmax[1])
JSONWindmaxLong=MB.JSONWIModelGrid(forecastTmax2[1],StartTmax2,EndTmax2,GridDefinitionTmax2[1],descriptionTmax2[1],intervalTmax2[0],UnitsTmax2[1])

MB.FTPUpload(host,Username,Password,filelocation=ModelDayStr,**filename=['TmaxShortTerm.json','TmaxLongTerm.json','TminShortTerm.json','TminLongTerm.json','PminShortTerm.json','PmaxShortTerm.json','PlikelyShortTerm.json','PminLongTerm.json','PmaxLongTerm.json','PlikelyLongTerm.json','WindmaxShortTerm.json','WindmaxLongTerm.json'],data=[JSONTmax,JSONTmaxLong,JSONTmin,JSONTminLong,JSONPmin,JSONPmax,JSONPlikely,JSONPminLong,JSONPmaxLong,JSONPlikelyLong,JSONWindmax,JSONWindmaxLong]**)
print('FTPUpload complete')

F.2 Uitwerking in Model Builder v2

```
import datetime
import numpy as np
import ModelBuilderv2 as MB

Today=datetime.datetime.now()
ModelDay=Today+datetime.timedelta(-1)
ModelDay=ModelDay.replace(hour=12,minute=0)
StartDate1=ModelDay+datetime.timedelta(1)
StartDate2=ModelDay+datetime.timedelta(4)
EndDate1=ModelDay+datetime.timedelta(4)
EndDate2=ModelDay+datetime.timedelta(10)
ModelDayStr=MB.dateFormat(ModelDay)

DataSetTW=MB.returnGridData("ModelGrid","CommonSense.Ethiopia.Ecmwf.Deterministic.6h",["ModelDate+StartDate",ModelDay,StartDate1,EndDate1],["TMAX","InstantaneousWindGust"],["Days","1"],"errormail",Name=["TmaxShortTerm.json","WindmaxShortTerm.json"],CalculationType="Max")
DataSetTWLong=MB.returnGridData("ModelGrid","CommonSense.Ethiopia.Ecmwf.Deterministic.6h",["ModelDate+StartDate",ModelDay,StartDate2,EndDate2],["TMAX","InstantaneousWindGust"],["Days",str((EndDate2-
StartDate2).days)],"errormail",Name=["TmaxLongTerm.json","WindmaxLongTerm.json"],CalculationType="Max")
DataSetTmin=MB.returnGridData("ModelGrid","CommonSense.Ethiopia.Ecmwf.Deterministic.6h",["ModelDate+StartDate",ModelDay,StartDate1,EndDate1],["TMIN"],["Days","1"],"errormail",Name=["TminShortTerm.json"],CalculationType="MIN")
DataSetTminLong=MB.returnGridData("ModelGrid","CommonSense.Ethiopia.Ecmwf.Deterministic.6h",["ModelDate+StartDate",ModelDay,StartDate2,EndDate2],["TMIN"],["Days",
str((EndDate2-
StartDate2).days)],"errormail",Name=['TminLongTerm.json'],CalculationType="MIN")
DataSetP=MB.returnGridData("EnsembleGrid","CommonSense.Ethiopia.Ecmwf.Eps.6h",["ModelDate+StartDate",ModelDay,StartDate1,EndDate1],["P"],["Days","1"],"errormail",Name=['PminShortTerm.json','PmaxShortTerm.json','PlikelyShortTerm.json'])
DataSetPLong=MB.returnGridData("EnsembleGrid","CommonSense.Ethiopia.Ecmwf.Eps.6h",["ModelDate+StartDate",ModelDay,StartDate2,EndDate2],["P"],["Days",str((EndDate2-StartDate2).days)],"errormail",Name=['PminLongTerm.json','PmaxLongTerm.json','PlikelyLongTerm.json'])

Pminpercent=30/100.
minind=int(np.floor((np.shape(DataSetP.Data[0,1:50,:,:,:])[0])*Pminpercent))
forecast_pmin1=np.sort(DataSetP.Data[0,1:50,:,:,:],0)[minind]
minind=int(np.floor((np.shape(DataSetPLong.Data[0,1:50,:,:,:])[0])*Pminpercent))
forecast_pmin2=np.sort(DataSetPLong.Data[0,1:50,:,:,:],0)[minind]

Pmaxpercent=90/100.
maxind=int(np.floor((np.shape(DataSetP.Data[0,1:50,:,:,:])[0])*Pmaxpercent))
```

```
forecast_pmax1=np.sort(DataSetP.Data[0,1:50,:,:,:],0)[maxind]
maxind=int(np.floor((np.shape(DataSetPLong.Data[0,1:50,:,:,:])[0])*Pmaxpercent))
forecast_pmax2=np.sort(DataSetPLong.Data[0,1:50,:,:,:],0)[maxind]

pthres=0.5
forecast_plikely1=np.zeros(np.shape(DataSetP.Data[0,1:50,:,:,:])[1:])

for i in range(len(DataSetP.Data[0,0,:,0,0])):
   for la in range(len(DataSetP.Data[0,0,0,:,0])):
      for lo in range(len(DataSetP.Data[0,0,0,0,:])):

forecast_plikely1[i,la,lo]=len(DataSetP.Data[0,1:50,i,la,lo][DataSetP.Data[0,1:50,i,la,lo]>=pt
hres])/float(len(DataSetP.Data[0,1:50,i,la,lo]))*100

forecast_plikely2=np.zeros(np.shape(DataSetPLong.Data[0,1:50,:,:,:])[1:])

for la in range(len(DataSetPLong.Data[0,0,0,:,0])):
   for lo in range(len(DataSetPLong.Data[0,0,0,0,:])):

forecast_plikely2[0,la,lo]=len(DataSetPLong.Data[0,1:50,0,la,lo][DataSetPLong.Data[0,1:50
,0,la,lo]>=pthres])/float(len(DataSetPLong.Data[0,1:50,0,la,lo]))*100

DataSetTW.JSONWIModelGrid()
DataSetTWLong.JSONWIModelGrid()
DataSetTmin.JSONWIModelGrid()
DataSetTminLong.JSONWIModelGrid()
DataSetP.JSONWIModelGrid([forecast_pmin1,forecast_pmax1,forecast_plikely1])
DataSetPLong.JSONWIModelGrid([forecast_pmin2,forecast_pmin2,forecast_plikely2])

MB.FTPUpload(host,UserName,Password,ModelDayStr,[DataSetTW,DataSetTWLong,Da
taSetTmin,DataSetTminLong,DataSetP,DataSetPLong])
print("FTP Upload complete")
```

Bijlage G Verificatie

Hierin worden de gekozen eisen even kort behandeld en genoemd waarom deze wel of niet voltooid zijn. Daarbij wordt soms verwezen naar de code en soms naar algemene instellingen of andere documenten.

Wat	Eis/Wens	Wie?	Stukje Code
Model builder gebruikt python scripts	Eis	WI	1)
Model builder heeft goed documentatie	Eis	Iedereen	2)
Model builder kan zelfstandig ECMWF data inlezen en gebruiken	Eis	WI	3)
Model builder is in staat gegevens op te halen	Eis	Business	3)
Werkt met alle data soorten	Wens	Iedereen	4)
Model Builder kan automatisch op dagelijkse interval worden uitgevoerd	Eis	WI	3)
Analyse moet live draaien of periodiek	Wens	Business	3)
Complexere analyses mogelijk maken	Eis	Bedrijfsvisie	-
Model builder draait scripts op geplande momenten	Wens	Business	3)
Model builder functioneert buiten HydroNET (ook bronnen buiten HydroNET)	Wens	Potentiële gebruiker	3)
Model builder is bruikbaar intern en partners	Wens	Bedrijfsvisie	-
Model builder is in staat gegevens te bewerken	Eis	Business	1)
Modelbuilder maakt analyses beheersbaar	Eis	Potentiële gebruiker	1)
Model builder kan ook andere data inlezen op keuze van de gebruiker	Wens	WI	4)
Model builder moet de uitkomsten op een ftp server kwijt kunnen	Wens	WI	5)
Gebaseerd op bestaande ideeën	Optie	Business	6)

1) De volledige model builder is geschreven in python
2) Er zijn opmerkingen in de model builder en er is een losse documentatie
3) De model builder draait op een task scheduler en download zelf de data
4) Er is een functie voor het ophalen van data, deze functie is zo opgesteld dat de data keuze is vrij is en enkel andere input parameters vereist, zie Bijlage C.2
5) Er is een ftp upload module, zie Fig. 4 op pagina 15.
6) De modelbuilder maakt gebruik van bestaande opties binnen HydroNET daarnaast is deze grotendeels gebaseerd op voorbeeldcodes voor kleine stappen via de handige links.