SommenGenerator WaterModellen

Gebruikershandleiding versie 2.1.0

HKV LIJN IN WATER

Opdrachtgever

Deltares
Enabling Delta Life

National Institute for Public Health and Environment
Ministry of Health, Welfare and Sports

Deltares en SSC Campus

SommenGenerator WaterModellen

Gebruikershandleiding versie 2.1.0

Auteur
Bart Thonus

PR3597
december 2023

Leeswijzer

Dit document bevat de gebruikershandleiding van de SommenGenerator WaterModellen die is ontwikkeld in opdracht van SSC Campus/Rijkswaterstaat WVL en in beheer is bij Deltares onder het SITO-programma[1]. Deltares heeft het beheer, onderhoud en de doorontwikkeling van de SommenGenerator WaterModellen uitbesteed aan HKV, de ontwikkelaar van de SommenGenerator WaterModellen. De SommenGenerator WaterModellen wordt in dit document aangeduid met de afkorting SGWM.

SGWM is een applicatie waarmee grote hoeveelheden invoer(bestanden) en rekenopdrachten geautomatiseerd aangemaakt kunnen worden voor verschillende soorten rekenmodellen (watermodellen). SGWM is generiek toepasbaar en geschikt voor de diverse watermodellen zoals Sobek, Waqua, Swan, D-Hydro en Hydra-NL. SGWM is beschikbaar als standalone executable die op elk computersysteem met een Windows besturingssysteem gebruikt kan worden. Daarnaast wordt SGWM standaard aangeboden in de werkruimte van het ModellenPlatform van SSC-Campus[2] als onderdeel van het NWM[3]. SGWM is dan onderdeel van een totaaloplossing waarbij ook gebruik gemaakt kan worden van een rekengrid (tegenwoordig zowel voor Linux als Windows) voor het uitvoeren van de modelberekeningen. SGWM kan ook gebruikt worden in combinatie met een 'eigen' rekencluster of rekengrid.

Vanaf SGWM versie 2.0.0 is de functionaliteit van SGWM voor MHWp5 (Maatgevend HoogWater processor versie 5) geïntegreerd in SGWM. Beide modules gebruiken dezelfde basisfunctionaliteit, met als belangrijkste verschil dat SGWM voor MHWp5 gebruikt maakt van een modellentrein in plaats van een enkel rekenmodel, zoals het geval is bij SGWM. Vanaf versie 2.0.0 worden beiden aangeduid met SGWM.

Bij het gebruik van SGWM en het lezen van (de voorbeelden in) deze handleiding is kennis verondersteld de gebruikte watermodellen (Sobek, Waqua, Swan, D-Hydro en Hydra-NL). Voor gebruik van SGWM op het NWM is kennis verondersteld van het ModellenPlatform van SSC-Campus en het gebruik van rekenclusters en rekengrids in combinatie met LSF (Load Sharing Facility).

Deze handleiding beschrijft hoe SGWM gebruikt kan worden. SGWM is een generieke oplossing voor het aanmaken van grote hoeveelheden modelberekeningen van een willekeurig rekenmodel. SGWM moet hiertoe worden geconfigureerd met behulp van een zogenoemd project-definitie-bestand (afgekort met PDB), een module-definitie-bestand (afgekort met MDB) en een set templatebestanden. Een project-definitie-bestand bevat de definitie van een sommenset waarvoor invoerbestanden moeten worden aangemaakt

[1] SITO staat voor 'Subsidieregeling Instituten voor toegepast onderzoek. Voorheen viel het beheer onder KPP wat staat voor 'Kennis voor Primaire Processen'.
[2] Deze werkruimte is aan te vragen bij Rijkswaterstaat WVL.
[3] NWM staat voor National Water Model.

en algemene instellingen. Het module-definitie-bestand bevat rekenmodel specifiek invoer, waaronder de definitie van templates en rekeninstellingen.

Deze handleiding start (in hoofdstuk 1) met een toelichting op het installeren/opstarten van SGWM op een willekeurige computer met een Windows besturingssysteem alsook binnen het NWM, in een werkruimte van het ModellenPlatform. Daarna volgt (in hoofdstuk 2) een uitgebreide toelichting op het configureren van het PDB, MDB en templatebestanden. Hierbij worden zowel fictieve voorbeelden als voorbeelden van bestaande rekenmodellen en modelschematisaties gebruikt (aangeduid als user examples). Het doel van dit hoofdstuk is het eigen maken van de configuratie mogelijkheden binnen SGWM, opdat u, als gebruiker, zelf aan de slag kunt met uw eigen configuraties en het gebruik van SGWM. In hoofdstuk 3 wordt de functionaliteit van SGWM (focus op de werking van de gebruikersschil van SGWM) stap voor stap toegelicht. Tot slot worden in hoofdstuk 4 specifieke verschillen tussen een SGWM-project en een MHWp5 project toegelicht.

De eerste versie van SGWM is ontwikkeld in 2017. Sinds die eerste versie, zijn er doorontwikkelingen geweest. De uitbreidingen en aanpassingen zijn opgenomen in de SGWM release notes. Let op: bij sommige doorontwikkelingen zijn aanpassingen noodzakelijk aan al bestaande SGWM configuraties. Deze aanpassingen zijn per versie, beschreven in de release notes van SGWM.

Inhoud

1 Installatie en opstarten SGWM

SGWM wordt beschikbaar gesteld in de werkruimte van het ModellenPlatform (onderdeel van het NWM) of kan op een eigen computer (PC of laptop) met een Windows besturingssysteem worden gebruikt. Wanneer SGWM in de werkruimte van met ModellenPlatform wordt gebruikt is de installatie van SGWM al uitgevoerd en kan deze via het Windows startmenu direct gestart worden; zie voor een toelichting paragraaf 1.2. De installatie en het gebruik van SGWM op een eigen computer is beschreven in paragraaf 1.1.

1.1 Installatie en gebruik SGWM op eigen computer

SGWM is ontwikkeld in Python. SGWM wordt voor gebruik aangeboden als executabel. In deze executabel zijn alle benodigde Python bibliotheken opgenomen, wat het gebruik vereenvoudigd[4]. Om SGWM te kunnen gebruiken volstaat het kopiëren van de SGWM executable naar het eigen computersysteem. Na het activeren van de SGWM executable wordt een soort tijdelijke Python werkruimte ingericht waarbinnen SGWM draait. Het opstarten van SGWM duurt daarom enkele seconden. Zodra de SGWM-gebruikersschil verschijnt is SGWM klaar voor gebruik. Zie hoofdstuk 3 voor het verder gebruik van SGWM.

Om gebruik te kunnen maken van SGWM op een eigen computersysteem zijn onderstaande (minimale en aanbevolen) systeemeisen van toepassing.

Tabel 1 Systeemeisen SGWM op eigen computer

Onderdeel	Minimaal	Aanbevolen
Processor	2.0 GHz	2.0+ GHz, 2+ cores
RAM	8.0 GB	8.0 GB
Vrije schijfruimte	5.0 GB	10.0 GB
Internetverbinding	Niet	Wel
Besturingssysteem	Windows 10 of recenter	Windows 10

1.2 Gebruik SGWM in NWM-omgeving

Het Nationaal Water Model (NWM) wordt door SSC-Campus gehost in een online platform, het ModellenPlatform. Gebruik van het NWM (en SGWM) in dit platform kan aangevraagd worden bij de beheerder van het NWM (zie ook https://www.helpdeskwater.nl/onderwerpen/applicaties-modellen/applicaties-per/watermanagement/watermanagement/nationaal-water-model/contact-opnemen/).

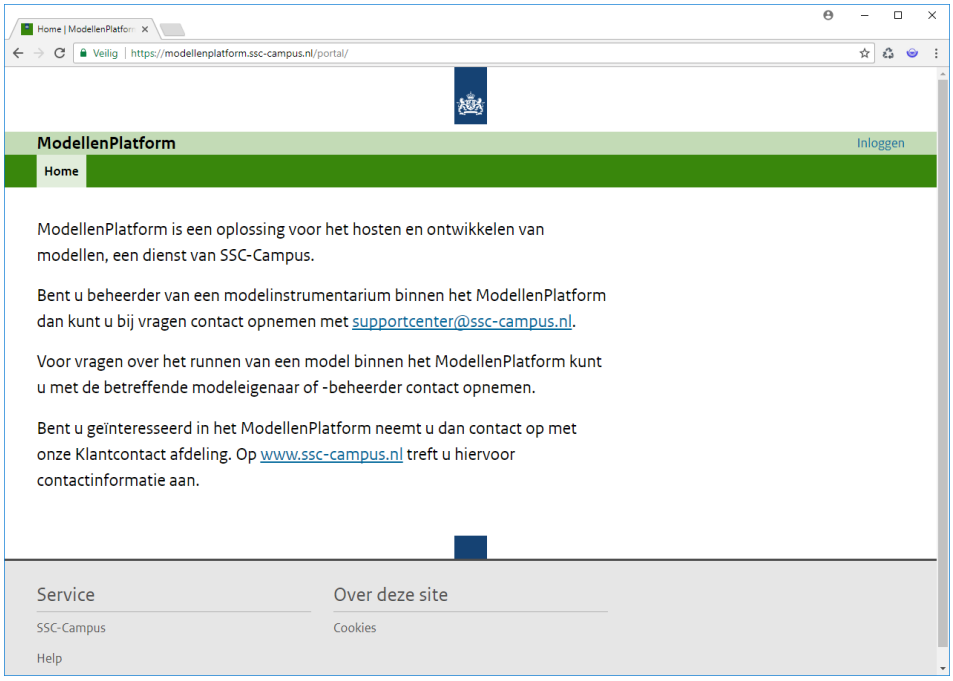
[4] Gebruik zonder ingewikkelde Python installatie.

In deze paragraaf wordt een beknopte toelichting gegeven over het gebruik van het ModellenPlatform voor het starten en gebruiken van SGWM. Voor aanvullende informatie over algemene gebruik van het ModellenPlatform en het gebruik voor het uitvoeren van rekenopdrachten wordt verwezen naar de documentatie van het ModellenPlatform (via SSC Campus).

Aanmelden op ModellenPlatform

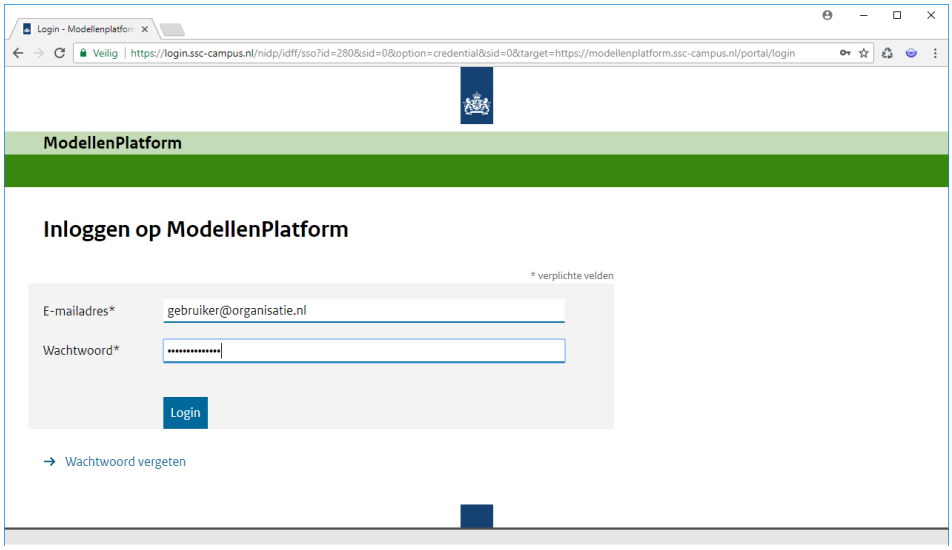
Open een browser en navigeer naar de startpagina van het ModellenPlatform: https://modellenplatform.ssc-campus.nl/portal/.

Figuur 1 Startpagina ModellenPlatform SSC-Campus



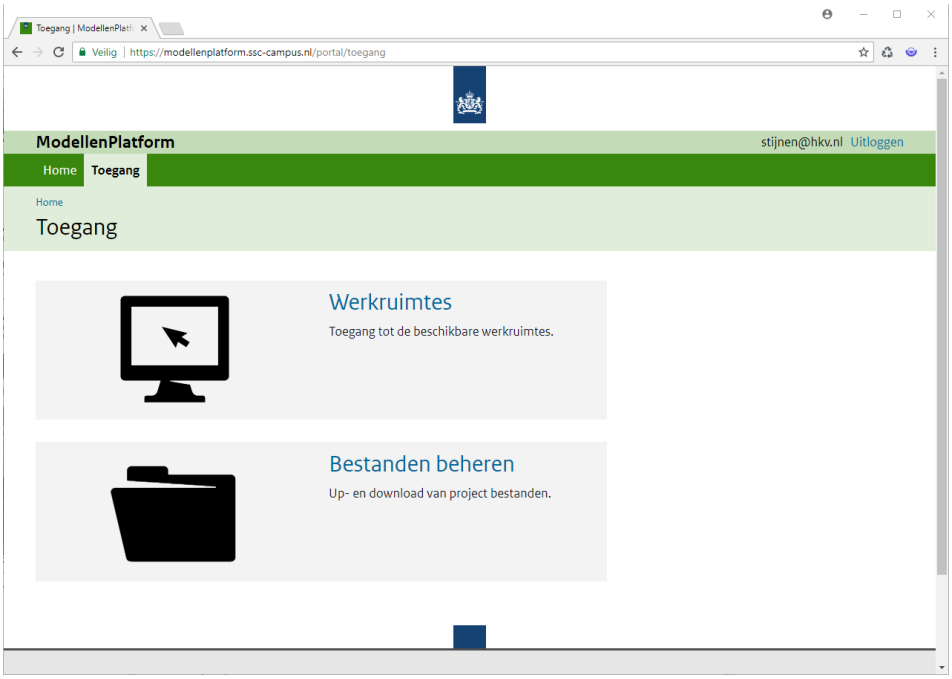
Activeer de link 'Inloggen' rechtsboven in het scherm. Gebruik de, door de beheerder verstrekte gegevens, voor het inloggen op het ModellenPlatform.

Figuur 2 Inloggen in ModellenPlatform



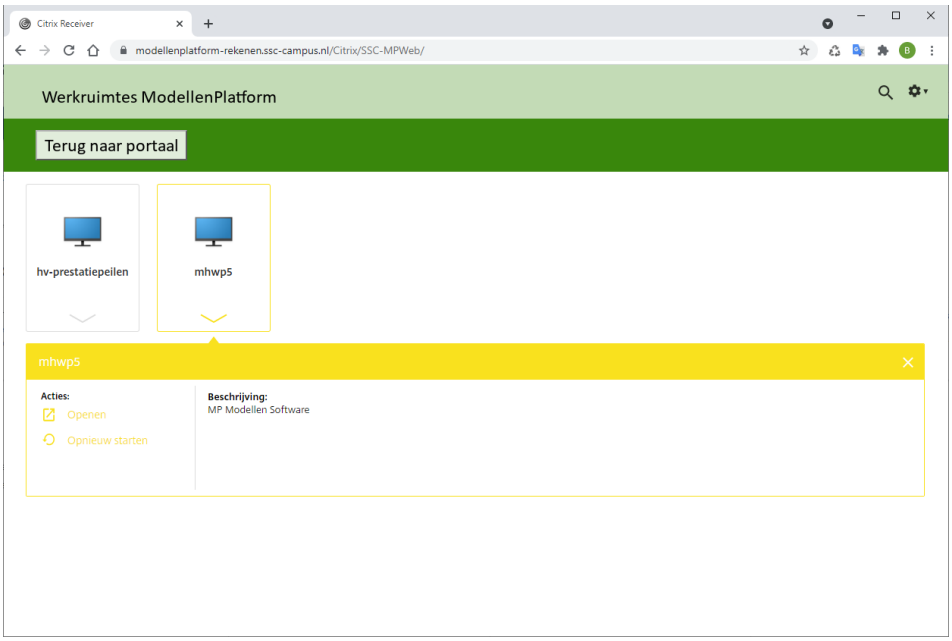
Activeer het tabblad 'Toegang' en klik op het blok 'Werkruimtes' om een virtuele werkruimte (Windows-omgeving) te starten.

Figuur 3
Starten virtuele werkruimte



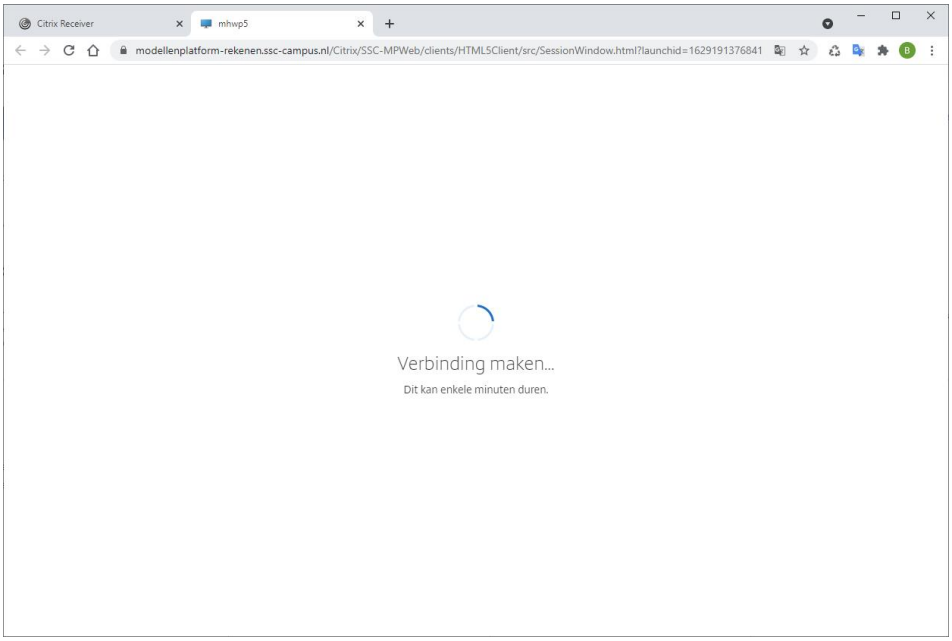
Bij het eerste gebruik kan mogelijk gevraagd worden om een Citrix plugin te installeren. Daarna verschijnt een overzicht met de beschikbare werkruimtes. Klik op het pijltje naar onder in een 'werkruimte icoon' om aanvullende informatie op te vragen. Klik op een 'werkruimte icoon' of de link 'Openen' om de werkruimte op te starten. De werkruimte wordt geopend (als Citrix omgeving) in een nieuw tabblad van de browser.

Figuur 4
Open het ICA-bestand



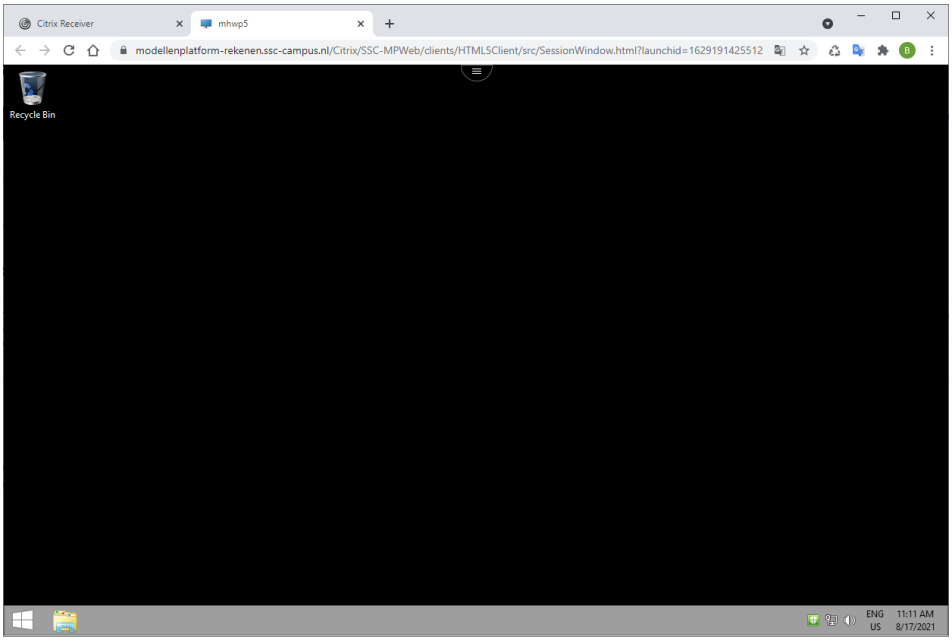
Er wordt een verbinding met de virtuele werkruimte tot stand gebracht.

Figuur 5 Verbinding maken met virtuele werkruimte



Als de verbinding gereed is dan verschijnt een Windows omgeving van de werkruimte in een apart browservenster.

Figuur 6 Windows werkruimte



Opstarten SGWM vanuit een virtuele werkruimte

Activeer het Windows startmenu in de virtuele werkruimte en klik op het programma ‘SGWM’. De gebruikersschil van SGWM verschijnt na enkele seconden en is klaar voor gebruik. Zie voor het verdere gebruik van SGWM, hoofdstuk 3.

2 SGWM-configuratie

2.1 Inleiding

SGWM heeft tot doel om, geautomatiseerd, invoerbestanden van een (willekeurig) rekenmodel klaar te zetten voor grote hoeveelheden modelsommen. Welke bestanden dit zijn varieert per (water)model en is onderdeel van een configuratie. Ook welke modelsommen beschouwd moeten worden is onderdeel van de configuratie van SGWM. We illustreren de globale werkwijze aan de hand van een beknopt (fictief) voorbeeld.

Stel men beschikt over een waterbewegingsmodel waarmee afvoeren en waterstanden op een rivier in een specifiek gebied kunnen worden berekend. Een dergelijk watermodel wordt aan de randen gevoed met zogenoemde randvoorwaarden, bijvoorbeeld de afvoer bovenstrooms en een zeewaterstand benedenstrooms. Vaak is men geïnteresseerd in de uitkomsten van een dergelijk model waarbij de randvoorwaarden (afvoer en zeewaterstand) variëren. Zo ontstaat er een tabel met bijvoorbeeld waterstanden op één of meerdere punten in de rivier als functie van 3 verschillende afvoerniveaus en 2 verschillende zeewaterstanden. Een dergelijk modeluitkomst kan dan bijvoorbeeld weer gebruikt worden in een probabilistisch rekenmodel om de faalkans van waterkeringen of de kans dat een bepaalde waterstand wordt overschreden, te berekenen.

Voor dit voorbeeld zou men SGWM willen configureren zodanig dat er 6 mappen worden aangemaakt met model invoerbestanden, één map voor elke combinatie van de afvoer en zeewaterstand (3 maal 2 is gelijk aan 6). Afvoer en zeewaterstand zijn dan de parameters in SGWM die gevarieerd worden in de verschillende sommen. Het waterbewegingsmodel maakt gebruik van een invoerbestand waarin de waarde van de afvoer en zeewaterstand zijn gedefinieerd. Van dit bestand maken we voor SGWM een template, waarin we de waarde van de afvoer en de zeewaterstand vervangen door zogenoemde SGWM Keys. SGWM maakt dan voor elke som een aparte map aan (met een unieke naam) met daarin een kopie van het templatebestand, waarin de SGWM Keys worden vervangen door de betreffende waarde van de afvoer en zeewaterstand van de specifieke som. Dus voor de som met een afvoer van 1000 m/s^3 en een zeewaterstand gelijk aan 2 m+NAP ontstaat een invoerbestand waarin deze waarden zijn ingevuld en voor de som met een afvoer van 2000 m/s^3 en een zeewaterstand gelijk aan 3 m+NAP ontstaat een invoerbestand waarin juist die waarden zijn ingevuld.

Bovenstaande voorbeeld beschrijft het primaire doel van SGWM waarbij de opzet van SGWM zo generiek mogelijk is, zodanig dat deze werkwijze past op verschillende watermodellen met elk hun eigen invoerbestanden en instellingen.

Dit hoofdstuk beschrijft de SGWM-configuratie aan de hand van een aantal voorbeelden. Dit betreft zowel de configuratie van een zogenoemd project-definitie-bestand (vanaf nu aangeduid als PDB), het module-definitie-bestand (vanaf nu aangeduid als MDB) en het gebruik van templatebestanden en SGWM Keys. Afhankelijk van het type modelberekening worden verschillende configuraties gebruikt. Er worden ook voorbeelden weergegeven voor verschillende rekenmodellen.

2.2 SGWM Project-definitie-bestand (PDB)

2.2.1 Inleiding

Het PDB is de basis van een (nieuw) SGWM-project. In dit bestand zijn de belangrijkste definities van een SGWM-configuratie opgenomen. Het PDB is een XML-bestand en kan geopend en aangepast worden in een willekeurige tekst-editor. Geadviseerd wordt om gebruik te maken van een tekst-editor met XML-tag herkenning[5] of een XML-editor, wat de kans op fouten bij het aanpassen van een XML-bestand verkleint.

De XML van een PDB heeft een standaard structuur met verplichte en optionele velden. Elk veld in de XML is gedefinieerd met zogenoemde XML-tags. Stel dat we het veld 'Naam' definiëren in een XML dan ziet dat er als volgt uit: <Naam>Een willekeurige naam</Naam>. Een veld wordt dus eerst gedefinieerd met de naam van het veld tussen een 'kleiner dan' en 'groter dan' teken (de open-XML-tag), daarna volgt de waarde van het veld en het veld wordt afgesloten met de naam van het veld tussen een 'kleiner dan' en 'groter dan' teken, waarbij voor de naam een (forward) slash is toegevoegd (de sluit-XML-tag).

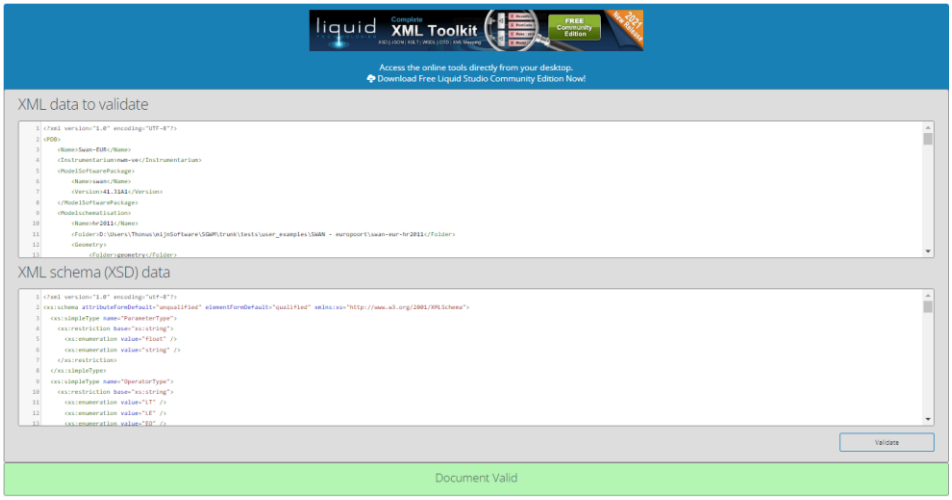
De hele definitie van de XML van een PDB ligt vast in een zogenoemd XSD-schema. Hierin zijn alle velden beschreven, is de hiërarchie tussen de velden aangeven en tot slot of een veld verplicht of optioneel is. Deze XSD-definitie wordt in SGWM gebruikt bij het inlezen van een PDB, ter controle op een correcte definitie. U kunt deze controle echter ook zelf uitvoeren voorafgaand aan het openen in SGWM. Gebruik hiervoor een XML-softwarepakket (bijvoorbeeld Altova XML; ook erg handig voor het aanpassen of aanmaken van een XML-bestand) of een online tool (bijvoorbeeld https://www.liquid-technologies.com/online-xsd-validator). Ter illustratie laten we zien hoe een PDB gevalideerd kan worden in een online tool.

Ga naar 'https://www.liquid-technologies.com/online-xsd-validator' en kopieer de inhoud van de XSD-definitie (zie bijlage G) naar het veld 'XML-schema (XSD) data'. Kopieer vervolgens de inhoud van de PDB naar het veld 'XML-data to validate'. Activeer de knop 'Validate' (na het aanvinken van de optie dat je geen robot bent). Het resultaat van de validatie wordt onderaan in het scherm weergeven. In Figuur 7 is het resultaat getoond van een valide

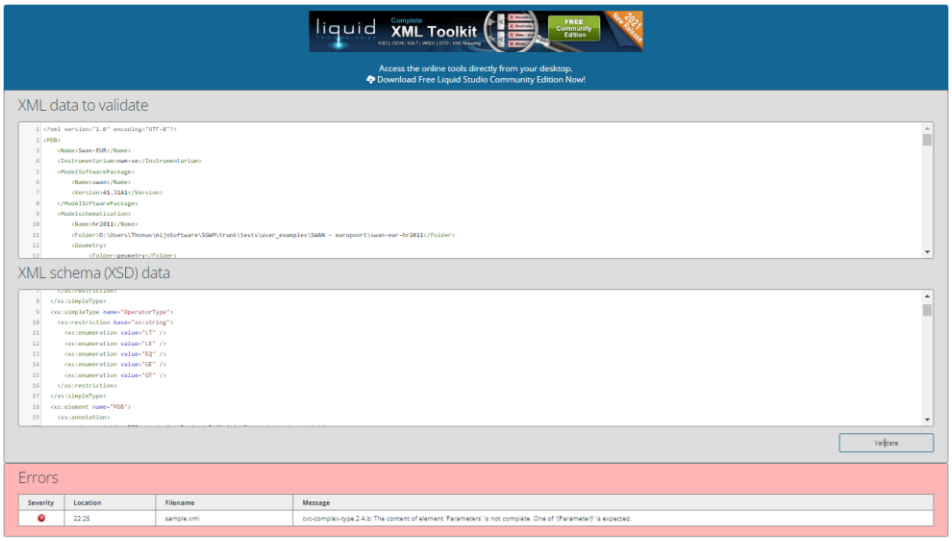
[5] Bijvoorbeeld Notepad++.

PDB en in Figuur 8 van een PDB waarin geen parameters zijn gedefinieerd. Dit wordt niet toegestaan, waardoor de foutmelding onderaan in het scherm wordt weergegeven.

Figuur 7
Online XSD-validatie van een valide PDB



Figuur 8
Online XSD-validatie van een foute PDB



2.2.2 De basis van een PDB

Een PDB begint en eindigt altijd met de XML-tag <PDB> (afkorting voor Project-Definitie-Bestand). Daarna volgen de kenmerken:

- Een willekeurige naam van het PDB aangeduid tussen de XML-tags <Name>;
- De naam van het instrumentarium dat gebruikt wordt om de modelberekeningen uit te voeren tussen de XML-tags <Instrumentarium>. Bij het gebruik in het kader van het NWM altijd gelijk aan 'nwm-ve'.

Beide zijn *verplichte* XML-tags die altijd opgenomen moeten worden in het PDB. Zowel de naam als het instrumentarium zijn metadata en worden verder niet gebruikt door SGWM bij het aanmaken van invoer voor sommen.

Een PDB bestaat daarna uit twee standaard blokken (beide *verplicht*), te weten:

1. Definitie van het gebruikte modelsoftwarepakket aangeduid met de XML-tags <ModelSoftwarePackage>;
2. Definitie van de modelschematisatie aangeduid met de XML-tags <Modelschematisation>;

Hieronder een voorbeeld van de basis van een PDB.

Basis XML van een PDB (allen verplichte tags)

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
        <Name>Voorbeeld</Name>
        <Instrumentarium>nwm-ve</Instrumentarium>
        <ModelSoftwarePackage></ModelSoftwarePackage>
        <Modelschematisation></Modelschematisation>
</PDB>
```

Let op dat elk item in een XML altijd begint met een tag tussen <> en afgesloten wordt met </>!

2.2.3 Definitie van een modelsoftwarepakket

In het PDB moet opgeven worden welk modelsoftwarepakket gebruikt voor de berekeningen. Deze informatie is metadata en wordt verder niet gebruikt door SGWM. Een modelsoftwarepakket wordt gekenmerkt door de naam van het modelsoftwarepakket, bijv. 'Waqua' (aangeduid met de XML-tags <Name>) en de versie-aanduiding van het modelsoftwarepakket, bijv. 'simona-2016-patch-09' (aangeduid met de XML-tags <Version>).

XML van een PDB met een modelsoftware-pakket (allen verplichte tags)

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
        <Name>Voorbeeld</Name>
        <Instrumentarium>nwm-ve</Instrumentarium>
        <ModelSoftwarePackage>
                <Name>software-naam</Name>
                <Version>software-versie-nummer</Version>
        </ModelSoftwarePackage>
        <Modelschematisation></Modelschematisation>
</PDB>
```

Beide tags zijn *verplichte* tags die altijd opgenomen moeten worden in een PDB.

2.2.4 Definitie van een modelschematisatie

Een modelschematisatie in een PDB bestaat uit een aantal standaard blokken:

1. Algemene kenmerken van de modelschematisatie, zoals de naam van de modelschematisatie (aangeduid met de XML-tags <Name>) en de naam van de rootfolder met alle bestanden behorende bij de modelschematisatie (aangeduid met de XML-tags <Folder>).
   Let op dat bij het uitwisselen van SGWM-projecten, de verwijzing naar de rootfolder met de bestanden van de modelschematisatie aangepast wordt aan een eventuele nieuwe situatie.

2. Een verwijzing naar de subfolder[6] met geometrische gegevens van de modelschematisatie aangeduid met de XML-tags <Geometry>.
3. Een verwijzing naar de subfolder[6] met randvoorwaarden gegevens van de modelsommen aangeduid met de XML-tags <BoundaryCondition>.
4. Een verwijzing naar de subfolder[6] met initiële conditie gegevens van de modelsommen aangeduid met de XML-tags <InitialCondition>.
5. Een definitie van sommen aangeduid met de XML-tags <Computation>.

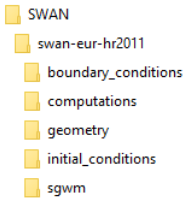
Bovengenoemde tags, met uitzondering van de XML-tags <Geometry>, <BoundaryCondition> en <InitialCondition> zijn verplichte tags in een PDB. Hieronder is een voorbeeld opgenomen van de standaard blokken van een modelschematisatie in een PDB.

XML van een PDB met een modelschematisatie

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
        <Name>Voorbeeld</Name>
        <Instrumentarium>nwm-ve</Instrumentarium>
        <Modelschematisation>
                <Name> voorbeeld-model-schematisatie</Name>
                <Folder>pad naar bestanden van model-
                        schematisatie</Folder>
                <Geometry>
                        <Folder>subfolder met geometrische
                                gegevens</Folder>
                </Geometry>
                <BoundaryCondition>
                        <Folder>submap met randvoorwaarden
                                bestanden</Folder>
                </BoundaryCondition>
                <InitialCondition>
                        <Folder> submap met initiële
                                condities</Folder>
                </InitialCondition>
                <Computation></Computation>
        </Modelschematisation>
</PDB>
```

In dit deel van een PDB zijn (relatieve) verwijzingen opgenomen naar bestanden van de modelschematisatie. Hieronder is een voorbeeld opgenomen van een bestandsstructuur zoals deze veel gebruikt wordt in watermodellen.

Figuur 9 Voorbeeld bestanden structuur modelschematisatie



[6] Altijd ten opzichte van de rootfolder met de modelschematisatie. N.B. in alle gevallen wordt deze verwijzing alleen gebruikt om te controleren of deze mappen fysiek bestaan. Het maakt ook niet uit welke mapnaam gekozen wordt of welke bestanden er in de mappen zijn opgenomen.

Er zijn mappen voor:

- bestanden met (model)randvoorwaarden in de map 'boundary_conditions';
- bestanden met geometrische (model)gegevens zoals bodemprofielen, ruwheden, etc. in de map 'geometry'[7];
- bestanden met initiële condities voor een model in de map 'initial_conditions'.

De map 'computations' wordt gebruikt voor het wegschrijven (door SGWM) van de invoer van de sommen en bevat ook de templatebestanden (zie ook paragraaf 2.4).
Tot slot bevat de map 'sgwm' de specifieke SGWM-invoerbestanden, zoals het PDB en MDB, maar bijvoorbeeld ook een projectbestand en een logbestand (zie verder ook hoofdstuk 3).

N.B. De naamgeving van de gebruikte mappenstructuur staat niet vast! Het is mogelijk om andere mapnamen te gebruiken (sinds SGWM-versie 1.1.3), bijvoorbeeld voor modelsommen waarbij een indeling in randvoorwaarden en initiële condities niet voor de hand ligt, zoals bij het rekenmodel Hydra-NL. Let wel dat deze namen dan ook aangepast worden in het PDB. SGWM controleert de aanwezigheid van de mappen in de directory structuur. **De naamgeving van de XML-tags (<Geometry>, <BoundaryCondition> en <InitialCondition>) kunnen niet gewijzigd worden, maar mogen eventueel wel weggelaten worden als er geen submappen nodig zijn.**

2.2.5 Definitie van sommen

Een modelschematisatie bevat een blok met daarin de definitie van sommen (ook wel 'computations' genoemd) die door SGWM moeten worden aangemaakt inclusief de benodigde model invoerbestanden per som. Deze model invoerbestanden worden door SGWM aangemaakt op basis van een set templatebestanden (zie ook 2.4).

Het blok met de definitie van sommen 'Computation' begint met een verwijzing naar de subfolder[6] (tussen de XML-tags <Folder>) waarin SGWM de mappen met model invoerbestanden per som moet gaan aanmaken. Dit is een verplicht veld in een PDB. Deze map is altijd relatief ten opzichte van de rootfolder zoals beschreven in paragraaf 2.2.4). Standaard wordt hier de mapnaam 'computations' voor gebruikt, maar dit mag ook een andere (map)naam zijn.
Daarna volgt de definitie van de zogenoemde somID (tussen de XML-tags <Id>). Een somID is de unieke naam die aan elke map met model invoerbestanden wordt gegeven door SGWM. **Omdat deze naam uniek moet zijn voor elke som moet minimaal elke SGWM Key van de gedefinieerde parameters opgenomen zijn in deze naam**.

[7] Dit zijn bestanden van de modelschematisatie die niet per som variëren.

Voorbeeld definitie van een somID

Uitgaande van het voorbeeld beschreven in paragraaf 2.1 beschouwen we een waterbewegingsmodel waarvoor 6 sommen aangemaakt moeten worden: 3 verschillende afvoerniveaus (500, 2.000 en 3.000 m^3/s) en 2 zeewaterstanden (2 en 3 m+NAP). Zowel de afvoer als de zeewaterstand beschouwen we in dit geval als parameter, immers voor beide worden meerdere verschillende waarden in de sommen gehanteerd. Voor de afvoer kiezen we de naam 'Afvoer' en de afkorting 'Q' die als SGWM Key kan worden gebruikt. Voor de zeewaterstand kiezen we de naam 'Zeewaterstand' en de afkorting 'L'. Het ligt voor de hand om de unieke somID aan de hand van deze alleen de SGWM Keys te definiëren, bijvoorbeeld '{Q}{L}'. Dit resulteert dan in de volgende mappen:

- Som1: 5002
- Som2: 5003
- Som3: 20002
- Som4: 20003
- Som5: 30002
- Som6: 30003

Voor de herkenbaarheid van de waarden uit de somID is het mogelijk om deze uit te breiden met willekeurige tekst (voor, tussen en na SGWM Keys), bijvoorbeeld 'Q{Q}L{L}'. Dit resulteert dan in de volgende mappen:

- Som1: Q500L2
- Som2: Q500L3
- Som3: Q2000L2
- Som4: Q2000L3
- Som5: Q3000L2
- Som6: Q3000L3

Maar elke willekeurige tekst kan worden toegevoegd aan de definitie van een somID, bijvoorbeeld 'som_ Q{Q}L{L}'. Dit resulteert dan in de volgende mappen:

- Som1: som_Q500L2
- Som2: som_Q500L3
- Som3: som_Q2000L2
- Som4: som_Q2000L3
- Som5: som_Q3000L2
- Som6: som_Q3000L3

In sommige gevallen is het wenselijk dat de lengte van een somID altijd gelijk is. Als dan gewerkt wordt met parameters waarvan de waarde niet altijd eenzelfde lengte hebben (bijvoorbeeld 500 en 3.000 m^3/s) dan is dat niet het geval. Sinds versie 2.1.0 biedt SGWM de mogelijkheid om het aantal karakters te benoemen in de somID. In het geval dat het aantal karakters van een waarde van een parameter niet overeenkomt met dit gedefinieerde aantal, wordt de overgebleven ruimte gevuld met zogenoemde voorloop nullen, bijvoorbeeld 'som_ Q{Q:4}L{L:2}'.

Dit resulteert dan in de volgende mappen:

- Som1: som_Q0500L02
- Som2: som_Q0500L03
- Som3: som_Q2000L02
- Som4: som_Q2000L03
- Som5: som_Q3000L02
- Som6: som_Q3000L03

Tot alle versies voor SGWM-versie 2.0.0, werd de somID altijd automatisch bepaald uit de definitie van de parameters (zie voor een beschrijving van parameters verderop in deze paragraaf). De volgorde die hierbij aangehouden werd was dan gelijk aan de volgorde van de definitie van de parameters in het PDB. Met de introductie van de XML-tag 'Id' is dit niet meer relevant. Ook is het mogelijk om variabelen (zie voor een beschrijving van variabelen verderop in deze paragraaf) op te nemen in de somID.

Daarna volgen in het blok 'Computation':

- Een blok, aangeduid met XML-tags <Parameters> waarin parameters worden gedefinieerd die variëren per som. De unieke combinatie van alle parameters en hun waarden, bepaalt de omvang van de sommenset in SGWM. De verschillende parameters kunnen als zogenoemde SGWM Key in templates gebruikt worden (zie ook paragraaf 2.4).
- Een blok, aangeduid met XML-tags <Variables> waarin aanvullende parameters (aangeduid met de term variabelen) gedefinieerd kunnen worden. De waarden van een variabele is altijd afhankelijk van de waarde van één of een combinatie van meerdere parameterwaarden. Ook de variabelen kunnen als zogenoemde SGWM Key in templates gebruikt worden (zie ook paragraaf 2.4).

XML van een PDB met de definitie van sommen (alle tags, m.u.v. <Variables> zijn verplicht)

```
<Computation>
        <Folder>de map waarin de invoer van de sommen door SGWM
              aangemaakt moet worden</Folder>
         <Id>Q_{Q:4}_L_{L:2}</Id>
        <Parameters>defintie van parameters</Parameters>
        <Variables>definitie van variabelen</Variables>
</Computation>
```

Definitie van parameters

Tussen de tag <Parameters> moeten de, in de sommen te variëren parameters, gedefinieerd worden. **Er moet altijd minimaal één parameter gedefinieerd worden**.

Elke parameter wordt gedefinieerd door een standaard blok.

XML van een PDB met een parameter (allen verplichte tags)

```
<Parameter>
        <Name>naam van een parameter</Name>
        <Key>SGWM Key naam voor gebruik in templatebestanden</Key>
        <Type>float of string</Type>
        <Values>
                <Value>waarde van een parameter</Value>
                …
                <Value>waarde van een parameter</Value>
        </Values>
</Parameter>
```

Een parameter wordt gekenmerkt door een willekeurige naam (aangeduid met de XML-tags <Name>) en een afkorting, de zogenaamde SGWM Key (aangeduid met de XML-tags <Key>). De naam van een parameter wordt getoond in de gebruikersschil van SGWM. De SGWM Key kan gebruikt worden in de verschillende templates bestanden (zie paragraaf ook 2.3).

In de parameterdefinitie moet aangegeven worden welke waarden de parameter kan aannemen in de verschillende sommen. Elke parameterwaarde wordt vastgelegd tussen de XML-tags <Value>. **Een parameter moet altijd minimaal één waarde bevatten.** Een parameterwaarde kan een tekst of een numerieke waarde (zowel een geheel als decimaal getal) zijn. In de parameterdefinitie moet aangegeven worden tussen de XML-tags <Type> of de waarden tekst zijn (type is dan gelijk aan 'string') of dat het numerieke waarden betreffen (type is dan gelijk aan 'float').

Voorbeeld parameter definitie

Uitgaande van het voorbeeld beschreven in paragraaf 2.1 beschouwen we een waterbewegingsmodel waarvoor 6 sommen aangemaakt moeten worden: 3 verschillende afvoerniveaus (1.000, 2.000 en 3.000 m^3/s) en 2 zeewaterstanden (2 en 3 m+NAP). Zowel de afvoer als de zeewaterstand beschouwen we in dit geval als parameter, immers voor beide worden meerdere verschillende waarden in de sommen gehanteerd. Voor de afvoer kiezen we de naam 'Afvoer' en de afkorting 'Q' die als SGWM Key kan worden gebruikt. De afvoeren zijn numerieke waarden, dus het type is gelijk aan een 'float'. De definitie van de eerste parameter is dan als volgt:

```
<Parameter>
        <Name>Afvoer</Name>
        <Key>Q</Key>
        <Type>float</Type>
        <Values>
                <Value>1000</Value>
                <Value>2000</Value>
                <Value>3000</Value>
        </Values>
</Parameter>
```

Voor de zeewaterstand kiezen we de naam 'Zeewaterstand' en de afkorting 'L'. Ook dit zijn numerieke waarden, dus het type is gelijk aan 'float'. De

definitie is dan als volgt:

```
<Parameter>
	<Name>Zeewaterstand</Name>
	<Key>L</Key>
	<Type>float</Type>
	<Values>
		<Value>2</Value>
		<Value>3</Value>
	</Values>
</Parameter>
```

Variabelen

SGWM biedt de mogelijkheid om naast parameters ook gebruik te maken van variabelen. Variabelen zijn, van één of meerdere parameters afgeleide waarden die gebruikt kunnen worden als SGWM Keys in de verschillende templates. De definitie van een variabele lijkt op die van een parameter, maar bepaalt dus niet de omvang van de sommenset zoals een parameter definitie dat wel doet!

Een variabele wordt gekenmerkt door een willekeurige naam (aangeduid met de XML-tags <Name>) en een SGWM Key (aangeduid met de tags <Key>). De SGWM Key van een variabele kan eveneens gebruikt worden in de verschillende templates. Daarnaast is net als bij een parameter, een type gedefinieerd (aangeduid met de XML-tags <Type>). Een type kan bestaan uit een 'string' voor tekstwaarden of een 'float' voor numerieke waarden (zowel een geheel als decimaal getal).

XML van een PDB met een variabele (allen verplichte tags)

```
<Variable>
	<Name>naam van een variabelen</Name>
	<Key>SGWM Key naam voor gebruik in templatebestanden</Key>
	<Type>float of string</Type>
	<Values>
		<Value></Value>
		…
		<Value></Value>
	</Values>
</Variable>
```

Een variabele kan meerdere waarden aannemen (net als een parameter), maar de waarde is altijd afhankelijk van de waarde van één (of meerdere) gedefinieerde parameter(s). Deze afhankelijkheid wordt aangeduid met de XML-tags <Value> binnen de XML-tags <Values>. De daadwerkelijke waarde van een variabele (aangeduid met de XML-tags <Value>) is afhankelijk van de waarde van (minimaal) één parameter. De definitie van afhankelijkheid is sinds versie 2.1.0 aangepast naar een compactere notatie en bestaat nu uit twee delen gescheiden door een puntkomma-teken met in het *linkerdeel* het *functievoorschrift* en *de waarde van een parameter* en in het *rechterdeel de waarde van de variabele*. Het functievoorschrift zelf bestaat uit *de parameter Key gevolgd door een operator en de waarde van de parameter*.

De operator kan bestaan uit (Let op: deze zijn gewijzigd na versie 2.1.0):

- == gelijk aan;
- <[8] kleiner dan;
- >[8] groter dan;
- <=[8] kleiner of gelijk aan;
- >=[8] groter of gelijk aan.

Complexe voorwaarden zijn nu ook mogelijk (bijvoorbeeld door twee voorwaarden van verschillende parameters te definiëren). Hiertoe kan gebruik gemaakt worden van de volgende toevoegingen in het functievoorschrift:

- and resultaat geldig als aan beide voorwaarden wordt voldaan;
- or resultaat geldig als aan één of beide voorwaarden wordt voldaan;
- not geldigheid van een voorwaarde omkeren.

Voorbeeld variabele definitie

Uitgaande van het voorbeeld beschreven in paragraaf 2.1 beschouwen we een variabele voor de status van een stuw, gelegen in één van de waterlopen uit het waterbewegingsmodel. Bij een afvoer van 2.000 m^3/s of lager is de stuw gesloten. Bij een afvoer hoger dan 2.000 m^3/s staat de stuw open. We definiëren nu een variabele 'Stuw' (aangeduid met de afkorting 'S') die twee (tekstuele) waarde kan aannemen, namelijk 'open' en 'gesloten', afhankelijk van de waarde van parameter 'Q'. Bij een Q kleiner dan of gelijk aan 2.000 m^3/s is de waarde van 'S' gelijk aan 'open' en anders gelijk aan 'gesloten'. We definiëren dus 2 ParameterValues voor 'S'.

```
<Variable>
	<Name>Stuw</Name>
	<Key>S</Key>
	<Type>string</Type>
	<Values>
		<Value>Q &lt;= 2000 ; open</Value>
		<Value>Q &gt;  2000 ; gesloten</Value>
	</Values>
</Variable>
```

Stel dat de stuw pas gesloten wordt bij een afvoer groter dan 2.000 m^3/s, maar dan alleen als de zeewaterstand lager is dan 2.4 meter. Dan krijgen we een complexer functievoorschrift:

```
<Variable>
	<Name>Stuw</Name>
	<Key>S</Key>
	<Type>string</Type>
	<Values>
		<Value>Q &lt;= 2000 ; open</Value>
		<Value>Q &gt;  2000 and L &gt;= 2.4 ; open</Value>
		<Value>Q &gt;  2000 and L &lt; 2.4 ; gesloten
		   </Value>
	</Values>
</Variable>
```

[8] Let op: het gebruik van '&' in deze notatie kan door sommige XML editors standaard gewijzigd worden in '&'.

Variabelen kunnen ook gebruikt worden om parameters die, om wat voor reden dan ook, als tekst gedefinieerd worden, om te zetten in numerieke waarden. Een voorbeeld is een meerpeil dat als tekst in de vorm van 'n040' ('n' van negatief) en 'p040' ('p' van positief) in de parameterdefinitie is opgenomen maar in een (hulp)variabele omgezet wordt naar een numeriek waarde, respectievelijk '-0.4' en '0.4'.

Voorbeeld variabele definitie (tekst naar numeriek)

```
<Parameter>
	<Name>Meerpeil</Name>
	<Key>M</Key>
	<Type>string</Type>
	<Values>
		<Value>m040</Value>
		<Value>p040</Value>
	</Values>
</Parameter>
<Variable>
	<Name>Meerpeil_numeriek</Name>
	<Key>Mnum</Key>
	<Type>float</Type>
	<Values>
		<Value>M == 'm040'; -0.4</Value>
		<Value>M == 'p040'; 0.4</Value>
	</Values>
</Variable>
```

2.3 SGWM Module-definitie-bestand (MDB)

2.3.1 Inleiding

Het MDB bevat rekenmodel specifieke invoer[9]. Het MDB is een XML-bestand en kan geopend en aangepast worden in een willekeurige tekst-editor. Geadviseerd wordt om gebruik te maken van een tekst-editor met XML-tag herkenning[10] of een XML-editor, wat de kans op fouten bij het aanpassen van een XML-bestand verkleint.

De XML van een MDB heeft een standaard structuur met verplichte en optionele velden. De hele definitie van de XML van een MDB ligt ook vast in een XSD-schema (net als bij het PDB). Hierin zijn alle velden beschreven, is de hiërarchie tussen de velden aangeven en tot slot of een veld verplicht of optioneel is. Deze XSD-definitie (zie bijlage G) wordt in SGWM gebruikt bij het inlezen van een MDB, ter controle op een correcte definitie. Deze controle kan ook handmatig uitgevoerd worden, zoals beschreven bij het PDB (zie paragraaf 2.2.1).

[9] In het geval dat SGWM voor MHWp5 wordt toegepast, wordt er een modellentrein van verschillende rekenmodellen beschouwd. Dan zal een SGWM-project meerdere MDB's bevatten. Zie ook hoofdstuk 4.
[10] Bijvoorbeeld Notepad++.

2.3.2 De basis van een MDB

Een MDB begint en eindigt altijd met de XML-tag <MDB> (afkorting voor Module-Definitie-Bestand). Daarna de willekeurige naam van het MDB aangeduid tussen de XML-tags <Name>[11]. Dit is een *verplichte* XML-tag die altijd opgenomen moeten worden in het MDB. De naam is (net als in het PDB) metadata en wordt verder niet gebruikt door SGWM bij het aanmaken van invoer voor sommen.

Een MDB bestaat daarna uit twee standaard blokken (allemaal *verplicht*), te weten:

1. Definitie van model specifieke instellingen voor elke som aangeduid met de tags <Computation>,
2. Definitie van rekeninstellingen aangeduid met de tags <CalculationSettings>.

Hieronder een voorbeeld van de basis van een MDB.

Basis XML van een MDB (allen verplichte tags)

```
<?xml version="1.0" encoding="UTF-8"?>
<MDB>
        <Name>Voorbeeld</Name>
        <Computation></Computation>
        <CalculationSettings></CalculationSettings>
</MDB>
```

Let op dat elk item in een XML altijd begint met een tag tussen <> en afgesloten wordt met </>!

2.3.3 Definitie (van model specifieke instellingen) van een som

Een computation in een MDB bestaat uit een aantal standaard blokken:

- Een blok, aangeduid met XML-tags <Templates> waarin verwijzingen opgenomen worden naar één of meerdere templatebestanden die als model invoerbestanden gebruikt worden per som. Deze template-bestanden kunnen SGWM Keys bevatten die afhankelijk van de betreffende som worden vervangen door de waarde van een parameter, een variabele of rekeninstelling (CalculationSetting; zie paragraaf 2.3.7).
- Een tweetal eigenschappen, tussen de XML-tags <Input> die bepalen of en wat de naam van een eventuele submap per som is, waarin de invoerbestanden worden aangemaakt en waarin het primaire model invoerbestand[12] wordt benoemd door SGWM.
- En tot slot een blok, aangeduid met de XML-tags <States>, waarin (minimaal) één of meerdere som-statussen gedefinieerd kunnen worden. Denk bijvoorbeeld aan wanneer is een som is aangemaakt, geslaagd, of wanneer een som mislukt is, etc. Deze status wordt tevens weergegeven in de gebruikersschil van SGWM (zie ook hoofdstuk 3).

[11] Logischerwijs is deze naam gelijk aan de naam van het PDB, zodat de twee bestanden aan elkaar gelinkt kunnen worden.
[12] Dit is het bestand wat een rekenmodel als invoer verwacht, bijvoorbeeld een invoer.hyd bestand van een Hydra berekening.

XML van een MDB met de definitie van sommen (alle tags zijn verplicht)

```
<Computation>
        <Templates>defintie van templates</Templates>
        <Input>naamgeving van mappen en invoerbestand</Input>
        <States>definitie van som statussen</States>
</Computation>
```

2.3.4 Definitie en gebruik van templates

Voor elke som moeten model invoerbestanden klaargezet worden door SGWM. Voor elk van deze invoerbestanden worden zogenoemde SGWM-templates aangemaakt. Welke model invoerbestanden dit zijn, is sterk afhankelijk van de gebuikte modelsoftware.

In een MDB moeten minimaal 3 templatebestanden gedefinieerd worden, te weten:

- Een templatebestand dat als model invoerbestand wordt gebruikt[13].
- Een templatebestand dat als jobscript wordt gebruikt.
- Een templatebestand dat als bsub commando script wordt gebruikt.

Om de verschillende soorten templatebestanden te herkennen is in SGWM een type template geïntroduceerd. Er zijn vier verschillende typen templates gedefinieerd:

1. Type gelijk aan 'inputfile': het templatebestand is het primaire model invoerbestand van het rekenmodel.
2. Type gelijk aan 'lsf': het templatebestand bevat het lsf jobscript waarmee een som wordt gestart.
3. Type gelijk aan 'bsub': het templatebestand bevat een bsub commando script waarmee een som kan worden toegevoegd aan de wachtrij van een rekenomgeving.
4. Type gelijk aan 'general': een willekeurig ander templatebestand, dat niet voldoet aan voorgaande definities.

De definitie van templatebestanden in het MDB begint met de definitie van de folder (tussen de XML-tags <Folder>), waarin de templatebestanden zijn opgeslagen[6] (zie ook paragraaf 2.3). Daarna volgt voor elk templatebestand een blok, waarin allereerst het type is gedefinieerd (tussen de XML-tags <Type>), gevolgd door de verwijzing naar het templatebestand[14] (tussen de XML-tags <File>) en een eigenschap die aangeeft of de inhoud van het templatebestand door SGWM bijgewerkt moet worden, tussen de XML-tags <Update>. Met het bijwerken van een templatebestand wordt hier bedoeld het vervangen van SGWM Keys die in de templatebestanden zijn gebruikt (zie ook paragraaf 2.3).

XML van een MDB met templates (allen verplichte tags)

```
<Template>
        <Type>type templatebestand</Type>
        <File>verwijzing naam templatebestand (naam)</File>
        <Update>bijwerken inhoud met SGWM Keys</Update>
</Template>
```

[13] Bijvoorbeeld voor Waqua een siminp-bestand, voor Swan een swn-bestand en voor D-Hydro een mdu-bestand.
[14] Deze verwijzing bestaat uit de naam en de extensie van het templatebestand.

Voorbeeld definitie template

Uitgaande van het voorbeeld beschreven in paragraaf 2.1 beschouwen we vier verschillende templatebestanden:

1. een model invoerbestand 'input.txt' dat gebruikt wordt in ons waterbewegingsmodel en waarin alle instellingen van een som zijn opgenomen. In ons voorbeeld zijn de waarde van de afvoer, de zeewaterstand en de stand van stuw belangrijk. Dit template bevat dus drie SGWM Keys die bijgewerkt moeten worden per som.
2. een lsf jobscripts 'lsf.job.script' waarin (DOS) uitvoer commando's staan voor het opstarten van de modelberekening. Dit templatebestand bevat eveneens SGWM Keys.
3. een bsub commando script 'bsub_file.sh' waarin de commando's staan om de sommen toe te voegen aan een rekencluster[15]. Dit template bevat eveneens SGWM Keys.
4. een bestand met specifieke instelling voor het rekencluster van het NWM 'options.sh'. Dit template bevat geen SGWM Keys.

De definitie van de templates in het PDB ziet er dan als volgt uit:

```
<Templates>
	<Template>
		<Type>inputfile</Type>
		<File>input.txt</File>
		<Update>true</Update>
	</Template>
	<Template>
		<Type>lsf</Type>
		<File>lsf.job.script</File>
		<Update>true</Update>
	</Template>
	<Template>
		<Type>bsub</Type>
		<File>bsub_file.sh</File>
		<Update>true</Update>
	</Template>
	<Template>
		<Type>general</Type>
		<File>options.sh</File>
		<Update>false</Update>
	</Template>
</Templates>
```

Het kan voorkomen dat een definitie nodig is waarbij afhankelijk van een specifieke waarde van een parameter of variabele, een ander (primair) model invoerbestand nodig is. In dat geval worden er dus meerdere templatebestanden gedefinieerd, allemaal van het type 'inputfile'. Aan deze definities moet een conditie toegevoegd worden waarin aangegeven wordt wanneer het ene of het andere inputbestand gebruikt moeten worden. We illustreren dit met een voorbeeld.

[15] In onze voorbeelden wordt gebruik gemaakt van een rekencluster met een Load Sharing Facility (LSF), zoals ook het geval is op het ModellenPlatform van het NWM.

Voorbeeld meerdere 'inputfile' templates met condities
Stel dat in ons voorbeeld uit paragraaf 2.1 voor een gesloten stuw een significant ander (primair) invoer modelbestand gebruikt moet worden dan bij een open stuw. We definiëren dan twee templatebestanden als (primair) invoer modelbestand), van het type 'inputfile'. De eerste moet gebruikt worden bij een open stuw en de tweede bij een gesloten stuw. De definitie van de templates in het PDB ziet er dan als volgt uit:

```
<Templates>
	<Template>
		<Type>inputfile</Type>
		<File>input_open.txt</File>
		<Conditions>
			<Condition>
				<Key>S</Key>
				<Value>open</Value>
				<Operator>EQ</Operator>
			</Condition>
		</Conditions>
		<Update>true</Update>
	</Template>

<Template>
		<Type>inputfile</Type>
		<File>input_gesloten.txt</File>
		<Conditions>
			<Condition>
				<Key>S</Key>
				<Value>gesloten</Value>
				<Operator>EQ</Operator>
			</Condition>
		</Conditions>
		<Update>true</Update>
	</Template>
	<Template>
		<Type>lsf</Type>
		<File>lsf.job.script</File>
		<Update>true</Update>
	</Template>
	<Template>
		<Type>bsub</Type>
		<File>bsub_file.sh</File>
		<Update>true</Update>
	</Template>
	<Template>
		<Type>general</Type>
		<File>options.sh</File>
		<Update>false</Update>
	</Template>
</Templates>
```

Het gebruik van condities in de definitie van templates is optioneel. Wanneer er geen verschillende versies van een template zijn dan wordt het blok met de definitie van een conditie (tussen de XML-tags <Conditions>) weggelaten.

Een conditie (aangeduid met de XML-tags <Condition>) bestaat uit de volgende verplichte eigenschappen:

1. De Key waarvoor de conditie wordt toegepast. Dit kan zowel een Key van een parameter als van een variabele zijn.
2. De waarde van de betreffende Key waaraan de conditie wordt getoetst (aangeduid met de tag <Value>). Dit kan afhankelijk van het type parameter of variabele een tekst of een numerieke waarde zijn.
3. Een operator van de conditie (aangeduid met de tag <Operator>). De volgende operators kunnen gebruikt worden:
   - EQ waarde is exact gelijk aan
   - LT waarde is kleiner dan[16]
   - LE waarde is kleiner dan of gelijk aan[16]
   - GT waarde is groter dan[16]
   - GE waarde is groter dan of gelijk aan[16]

Als er gebruik gemaakt wordt van een conditie zijn bovenstaande tags verplicht.

Conditie afhankelijk van een rekeninstelling

Het is ook mogelijk om een conditie te definiëren die niet afhankelijk is van de waarde van een parameter of een variabele, maar die afhankelijk is van de waarde van een zogenoemde 'calculationsetting' (een rekeninstelling). 'Calculationsettings' zijn nog niet toegelicht op dit punt in deze gebruikershandleiding (zie daarvoor paragraaf 2.3.7). In deze paragraaf wordt al wel toegelicht dat, met de toevoeging van een extra attribuut aan de XML-tag <Condition>, gedefinieerd kan worden dat de waarde van een rekeninstelling gebruikt moet worden als conditie voor een template. Het toevoegen van een attribuut in de XML ziet eruit als in onderstaand voorbeeld.

XML van een MDB met template definitie inclusief condition afhankelijk van een 'calculation setting'

```
<Template>
	<Type></Type>
	<File></File>
	<Conditions>
		<Condition calculationsetting="true">
			<Key></Key>
			<Value></Value>
			<Operator></Operator>
		<Condition>
	</Conditions>
	<Update></Update>
</Template>
```

Als de waarde van het attribuut 'calculationsetting' in de XML-tag <Condition> gelijk is aan 'true' dan wordt de conditie gecontroleerd op een rekeninstelling en is die gelijk aan 'false' dan wordt in de conditie gecontroleerd op de Keys van een parameter of variabele. Bij het weglaten van het attribuut 'calculationsetting', wordt de standaard instelling gehanteerd en die is gelijk aan 'false'. Het gebruik van de conditie op basis van een rekeninstelling is verder gelijk aan die van een parameter of variabele.

[16] Kan alleen gebruikt worden bij numerieke waarden; niet bij tekst (string).

Optioneel: toevoegen van een shell commando in een bsub template

Op dit punt in de handleiding hebben we de inhoud van het bsub bestand nog niet toegelicht (zie hiervoor de toelichting paragraaf 2.4). De inhoud van dit bestand wordt per som opgebouwd, door telkens voor elke som een LSF submit commando te voegen. Het templatebestand bevat dit commando (dat voor elke som gebruikt wordt). Echter in sommige gevallen is het noodzakelijk/gewenst om een bsub bestand te beginnen met een uniek shell commando. Als we dit zouden toevoegen aan het bsub template, zou dit commando voor elke som herhaald worden. Dit is niet gewenst. Daarom bevat het MDB de mogelijkheid om aan de XML-tag <File> een extra attribuut 'shell' toe te voegen, waarin dit 'begin-shell-commando' is gespecificeerd. De definitie van het bsub template ziet er dan als volgt uit.

XML van een MDB met extra attribuut voor eenmalig shell commando in het bsub bestand

```
<Template>
        <Type>bsub</Type>
        <File shell="#!/bin/bash">bsub_file.sh</File>
        <Update></Update>
</Template>
```

2.3.5 Definitie van model input

SGWM maakt model invoerbestanden aan voor elke som die gedefinieerd wordt in het PDB[17]. Voor elke som maakt SGWM een subfolder aan in de map 'computations'[18] waarin de invoer voor de betreffende som wordt weggeschreven[19]. De naam van de subfolder is al gedefinieerd in het PDB (tussen de XML-tags <Id>). In het MDB kan tussen de XML-tags <Folder> (binnen de XML-tags <Input>) eventueel nog een extra subfolder aangemaakt worden zodat, de door SGWM gegenereerde invoer, gescheiden wordt opgeslagen van de resultaten. Begin de definitie van deze subfolder nooit met een '/'. Dus in plaats van '/invoer' is de definitie gelijk aan 'invoer' of 'invoer/subfolder' in plaats van '/invoer/subfolder'. **Het is niet mogelijk om SGWM Keys te gebruiken in de definitie van de subfolder.**

XML van een MDB met instellingen voor aanmaken van folders met model invoer bestanden

```
<Input>
        <Folder>invoer</Folder>
        <File></File>
</Input>
```

Definitie van (primair) model invoerbestand

De huidige versie van SGWM verwacht **altijd minimaal één template-bestand met invoer voor het betreffende rekenmodel (van het type 'inputfile'**; zie ook pararaaf 2.3.4). Voor Waqua is dit bijv. een 'siminp-bestand' en voor D-Hydro een 'mdu-bestand'. De definitie van dit type templatebestanden (primair model invoerbestand) is al beschreven in paragraaf 2.3.4. Daar is ook beschreven dat er mogelijk meerdere versies

[17] Zoals beschreven in paragraaf 2.2.5 wordt het aantal sommen bepaald door het aantal gedefinieerde parameters en hun waarden in het PDB.
[18] Of andere map als deze anders gedefinieerd is in het PDB (zie paragraaf 2.2.5).
[19] En waarin ook de resultaten van een som door het rekenmodel weggeschreven zullen worden.

van deze templates kunnen worden onderscheiden. Deze zullen dan ook verschillende namen hebben (zoals in het getoonde voorbeeld in paragraaf 2.3.4: 'input_open.txt' en 'input_gesloten.txt'). Gewenst is echter dat de naam van het primaire model invoerbestand voor elke som gelijk is. Immers het rekenmodel zal altijd één en hetzelfde invoerbestand gebruiken. Daarom is in het MDB binnen de XML-tags <Input> ook een XML-tag <File> opgenomen, waarin de naam van het primaire model invoerbestand kan worden opgegeven, zoals deze dan ook wordt weggeschreven in de mappen per som.

Tussen de tags <File> moet een definitie opgegeven worden van de naam van het ingevulde primaire invoerbestand per som (die door SGWM in de folder van de betreffende som wordt gekopieerd). Onderstaand voorbeeld toont een definitie waarbij (bei)de template(s) van het primaire model invoerbestand wordt hernoemd in 'input' gevolgd door de unieke ID van de som. Het is dus mogelijk om hier SGWM Keys te gebruiken. Hier is het ook mogelijk om gebruik te maken van de optionele definitie van het aantal karakters zoals beschreven in paragraaf 2.2.5.

XML van een MDB met instellingen voor aanmaken van de naam van een primair modelinvoer bestand

```
<Input>
	<Folder>invoer</Folder>
	<File>input.Q{Q:4}H{H}.txt</File>
</Input>
```

Het blok <Input> en de tag <Folder> en <File> zijn verplichte onderdelen van elk MDB.

SGWM is generiek van opzet en kan in beginsel toegepast worden voor elk soort rekenmodel. Mocht het zo zijn dat voor een rekenmodel geen specifiek model invoerbestand wordt gebruikt, definieer dan een dummy template-bestand van het type 'inputfile' en vul bij de tag <File> ook een dummy naam in; zoals hieronder ter illustratie is weergegeven.

XML van een MDB met een voorbeeld van een dummy primair modelinvoer bestand

```
<Templates>
	<Template>
		<Type>inputfile</Type>
		<File>dummy.txt</File>
		<Update>false</Update>
	</Template>
</Templates>
<Input>
	<Folder>invoer</Folder>
	<File>dummy.txt</File>
</Input>
```

2.3.6 Definitie van resultaat statussen

Nadat SGWM alle invoerbestanden per som heeft aangemaakt, zou je in de tabel met het overzicht van alle sommen in de gebruikersschil (zie ook paragraaf 3.5), per som de status van de som gelijk willen hebben aan 'invoer klaar' of 'invoer gereed'. Vervolgens zal een gebruiker daadwerkelijk sommen gaan starten op een (willekeurige) rekenomgeving en is het nuttig om de status van deze sommen te kunnen monitoren in SGWM. Daartoe is

een generieke definitie van 'som statussen' toegevoegd aan SGWM (vanaf versie 2.0.0), waarin SGWM de status van een som ontleent aan (de inhoud van) één van de model uitvoerbestanden.

Deze (generieke) definitie is onderdeel van het laatste blok binnen de XML-tags <Computation> in het MDB en wordt aangeduid met de XML-tags <States>.

Voor elke status definitie wordt een XML-tag <State> toegevoegd waarin de volgende onderdelen gedefinieerd moeten worden:

- De naam/omschrijving van de status tussen de XML-tags <Name>. Dit is ook de naam die weergegeven zal worden in het overzicht van alle sommen in de gebruikersschil (zie 3.8). Dit onderdeel is verplicht.
- Een tekst uit een resultaatbestand van een modelberekening, op basis waarvan geconcludeerd kan worden dat de berekening voldoet aan de gedefinieerde status, tussen de XML-tags <Text>. Dit onderdeel is optioneel. De status kan ook alleen op basis van de aanwezigheid van bestanden gecontroleerd worden (zie voorbeelden hieronder).
- De naam van het resultaatbestand van een modelberekening, tussen de XML-tags <File>. Dit onderdeel is verplicht.
- De naam van de (sub)folder waarin het resultaatbestand wordt weggeschreven[20], tussen de XML-tags <Folder>. Dit onderdeel is verplicht mag ook leeg zijn.
- Of de voortgangsbalk in de gebruikersschil, bijgewerkt moet worden aan de hand van de gedefinieerde status. Een waarde 'true' tussen de XML-tags <Progress> geeft aan dat de voortgang wordt gemeten aan de hand van deze status. Een waarde 'false' geeft aan dat die status wordt genegeerd in de bepaling van de voortgang. **Er moet minimaal één status met 'Progress' gelijk aan 'true' gedefinieerd worden om voortgang weer te kunnen geven in de gebruikersschil van SGWM.**

Het is aan de gebruiker om statussen en de bijbehorende voorwaarden van een status te definiëren. Hieronder geven we een tweetal voorbeelden. **Er moet minimaal 1 state gedefinieerd worden in het MDB.**

Voorbeelden van een status definitie

Stel dat in het voorbeeld uit paragraaf 2.2.1 gebruik gemaakt wordt van een waterbewegingsmodel dat het resultaat van een geslaagd berekening wegschrijft in een bestand met de naam 'results.out'. Bij een berekening met fouten wordt er een extra uitvoerbestand aangemaakt met de naam 'Error.txt'. We zouden nu twee som statussen kunnen onderscheiden, namelijk:

- een geslaagde som met de status omschrijving/naam 'som gereed' en
- een som die niet geslaagd is met de status omschrijving/naam 'som mislukt'. N.B. de namen die hier voor de statussen worden gekozen, zijn willekeurig. Elke naamgeving is mogelijk.

[20] Let op! De referentie naar een folder is hier altijd ten opzichte van de folder van een som (meestal aangeduid met de som ID).

We weten nu dat als er een bestand Error.txt verschijnt in de map met resultaten, dat de som mislukt is. We kunnen hiervoor de volgende status in het MDB definiëren:

```
<State>
	<Name>som mislukt</Name>
	<Text></Text>
	<File>Error.txt</File>
	<Folder>result</Folder>
	<Progress>false</Progress>
</State>
```

We weten ook dat als het bestand 'Error.txt' er niet is, maar er is wel een bestand 'results.out', dat de som geslaagd is. We kunnen hiervoor een tweede status in het MDB definiëren:

```
<State>
	<Name>som gereed</Name>
	<Text></Text>
	<File>results.out</File>
	<Folder>result</Folder>
	<Progress>true</Progress>
</State>
```

Tot slot definiëren we ook een status voor de situatie waarin SGWM wel al invoerbestanden heeft aangemaakt, maar er nog geen sommen zijn gestart:

```
<State>
	<Name>invoer gereed</Name>
	<Text></Text>
	<File>input.file</File>
	<Folder>input</Folder>
	<Progress>false</Progress>
</State>
```

Als we de statussen in deze volgorde definiëren zal SGWM eerst controleren of er een bestand 'Error.txt' voorkomt in de map 'result'. Is dat het geval, dan krijgt de som de status 'som mislukt'. Als dat bestand er niet is dan controleert SGWM de aanwezigheid van het bestand 'results.out' in dezelfde map. Als die er is krijgt de som de status 'som gereed'. Wordt dat bestand ook niet gevonden, dan controleert SGWM of het bestand 'input.file' voorkomt in de map 'input'. Als dat het geval is, dan wordt de status gelijk aan 'invoer gereed'. Wordt dat bestand niet gevonden dan krijgt de som geen status.

Zouden we de volgorde van deze definitie omdraaien, dan zou de status 'som mislukt' nooit voorkomen omdat het model altijd een bestand 'results.out' aanmaakt (ook als de som mislukt). In dat geval wordt dus altijd voldaan aan de eerste voorwaarde en is de status altijd gelijk aan 'som gereed'. Let dus goed op de volgorde bij het definiëren van de statussen.

In het bovenstaande voorbeeld is de XML-tag <Text> niet gevuld in de definitie in het MDB. Stel dat een rekenmodel slechts één resultaatbestand

wegschrijft, dan zal de status uit de inhoud van dit bestand gelezen moeten worden. Dan moet ook de XML-tag <Text> gebruikt worden. Stel dat het rekenmodel alleen het bestand 'results.out' wegschrijft en bij een geslaagde som dit bestand altijd afsluit met de tekst 'Som succesvol geëindigd'. In dat geval kunnen we volstaan met onderstaande definitie van de statussen in het MDB:

```
<States>
	<State>
		<Name>som gereed</Name>
		<Text>Som succesvol geëindigd</Text>
		<File>results.out</File>
		<Folder>result</Folder>
		<Progress>true</Progress>
	</State>
	<State>
		<Name>som mislukt</Name>
		<Text></Text>
		<File>results.out</File>
		<Folder>result</Folder>
		<Progress>false</Progress>
	</State>
	<State>
		<Name>invoer gereed</Name>
		<Text></Text>
		<File>input.file</File>
		<Folder>input</Folder>
		<Progress>false</Progress>
	</State>
</States>
```

Let op dat bij het controleren van de aanwezigheid van specifieke tekst in een resultaatbestand, deze tekst hoofdlettergevoelig is.

Tot slot wordt opgemerkt dat sommige rekenmodellen een deel van de naam van het model invoerbestand gebruiken, in de naamgeving van de resultaat-bestanden. Aangezien wij eerder hebben beschreven dat we de naam van het invoerbestand afhankelijk kunnen laten zijn van de somID, is de naam van het resultaat bestand dat soms ook. Daarom hebben we de mogelijkheid toegevoegd aan SGWM om de naam van de bestanden in de definitie van de statussen (tussen de XML-tags 'File'), met een asterisk te definiëren. Zie het voorbeeld hieronder.

Gebruik van een asterisk in de naam van een uitvoerbestand
Beschouw onderstaande definitie van een status in een MDB.

```
<State>
	<Name>som gereed</Name>
	<Text></Text>
	<File>*.out</File>
	<Folder>result</Folder>
	<Progress>true</Progress>
</State>
```

In dit geval zal elke som de status 'som gereed' krijgen zodra er een willekeurig bestand met de extensie '.out' verschijnt in de folder met resultaatbestanden (in dit geval 'result').

2.3.7 Definitie van rekeninstellingen

In SGWM is gekozen voor een generieke definitie van rekeninstellingen. Rekeninstellingen zullen veelal verschillen per rekenmodel maar vooral ook per rekenomgeving, maar zijn daarna voor alle sommen gelijk. Rekeninstellingen kunnen gedefinieerd worden in het blok 'CalculationSetttings' in een MDB (aangeduid met de XML-tags <CalculationSettings>). **Er moet altijd minimaal één rekeninstelling gedefinieerd worden**.

XML van een module-definitie-bestand met projectinstellingen (de vetgedrukte tags zijn verplichte tags)

```
<CalculationSetting>
        <Name>naam rekeninstelling</Name>
        <Key>SGWM Key rekeninstelling</Key>
        <Type>float, integer, string, comobo, file of folder</Type>
        <Items>bij gebruik van combo de items uit de
                keuzelijst</Items>
        <Value>de default waarde</Value>
        <Edit>wel of niet aanpasbaar in de gebruikersschil</Edit>
</CalculationSetting>
```

Rekeninstellingen die in het MDB worden gedefinieerd, kunnen weergegeven (en de waarde daarvan aangepast) worden in de gebruikersschil van SGWM (zie ook paragraaf 3.5). De naam die weergegeven wordt in het scherm met rekeninstellingen in de gebruikersschil, wordt gedefinieerd tussen de XML-tags <Name>. Elke rekeninstelling kan ook als SGWM Key gebruikt worden in de templatebestanden. Deze werkwijze maakt het mogelijk dat de gebruiker in het scherm met rekeninstellingen van SGWM, een aanpassing van een waarde doet, die dan in de templates verwerkt wordt bij het aanmaken van de invoer van sommen. De Keys worden gedefinieerd tussen de XML-tags <Key>. Van elke rekeninstelling kan aangegeven worden van welke type de instelling is (tussen de XML-tags <Type>). De volgende mogelijkheden zijn geïmplementeerd:

- 'string' een willekeurige tekst die in een tekstveld wordt gepresenteerd;
- 'float' een decimale waarde die in een vrij invoerveld wordt gepresenteerd;
- 'integer' een geheel getal dat in een numeriek invoerveld wordt gepresenteerd;
- 'combo' een keuzelijst (strings) die in een keuzeveld wordt gepresenteerd;
- 'file' een verwijzing naar een bestand die een bestanddialoog kan worden geselecteerd;
- 'folder' een verwijzing naar een folder die een folderdialoog kan worden geselecteerd;

Tussen de XML-tags <Value> wordt de default waarde van een rekeninstelling opgenomen. Afhankelijk van het gekozen type rekeninstelling is dit een tekst of een numerieke waarde. Als gekozen wordt voor het type 'combo' dan moet een extra XML-tag opgenomen worden, namelijk <Items>. Hierin worden de verschillende items uit de keuzelijst gedefinieerd. Items zijn altijd tekst die gescheiden worden door een puntkomma.

De laatste tag in het blok met rekeninstellingen betreft <Edit> waarmee aangeduid kan worden of de rekeninstelling zichtbaar wordt gemaakt in de gebruikersschil of niet. Met de waarde 'true' kan de gebruiker de instellingen wijzigen in de gebruikersschil en met de waarde 'false' niet en wordt altijd de default waarde uit het MDB gebruikt in de templates.

Sinds SGWM versie 2.0.0 is het mogelijk om rekeninstellingen (net als parameters en variabelen) te gebruiken als SGWM Keys in templatesbestanden (zie ook paragraaf 2.4). Daarnaast is de mogelijkheid geïntroduceerd om te werken met zogenoemde conditionele rekeninstellingen. Het is dan mogelijk om een rekeninstelling afhankelijk van de waarde van een eerder gedefinieerde rekeninstelling te laten wijzigen. Als gebruiker heb je dan geen controle meer over de waarde van deze rekeninstelling, maar wijzigt deze 'mee' met het aanpassen van een andere rekeninstelling. Hieronder is een voorbeeld opgenomen van een conditionele rekeninstelling.

Gebruik conditionele rekeninstelling (calculationsetting)
Stel voor dat we twee rekeninstellingen nodig hebben voor ons voorbeeld uit paragraaf 2.1. In de eerste rekeninstelling definiëren twee verschillende klimaat scenario's: 2050 en 2100. De definitie van deze rekeninstelling is als volgt:

```
<CalculationSetting>
	<Name>Klimaatscenario</Name>
	<Key>Klimaat</Key>
	<Type>combi</Type>
	<Items>2050,2100</Items>
	<Value>2050</Value>
	<Edit>true</Edit>
</CalculationSetting>
```

Vervolgens willen we een toeslag voor de verwachtte zeespiegelstijging toe passen op de zeewaterstand in de berekeningen, afhankelijk van het gekozen klimaatscenario. We definiëren daarom een tweede conditionele rekeninstelling als volgt:

```
<CalculationSetting>
	<Name>Zeespiegelstijging</Name>
	<Key>ZSS</Key>
	<Type>float</Type>
	<Conditions>
		<Condition>
			<CalculationSettings>
				<CalculationSetting>
					<Key>Klimaat</Key>
```

```
                                    <Value>2050</Value>
                                    <Operator>EQ</Operator>
                              </CalculationSetting>
                        </CalculationSettings>
                        <Value>0.05</Value>
                  </Condition>
                  <Condition>
                        <CalculationSettings>
                              <CalculationSetting>
                                    <Key>Klimaat</Key>
                                    <Value>2100</Value>
                                    <Operator>EQ</Operator>
                              </CalculationSetting>
                        </CalculationSettings>
                        <Value>0.45</Value>
                  </Condition>
            </Conditions>
</CalculationSetting>
```

Bij een gekozen klimaatscenario 2050 wordt de zeespiegelstijging door SGWM automatisch gelijk gemaakt aan 5 cm en bij 2100 gelijk aan 45 cm.

2.4 SGWM-templates en SGWM Keys

In paragraaf 2.3 is het gebruik van templates al geïntroduceerd. Er zijn vier verschillende typen templates onderscheiden:

1. Type gelijk aan 'inputfile': het templatebestand is het primaire model invoerbestand van het rekenmodel.
2. Type gelijk aan 'lsf': het templatebestand bevat het lsf jobscript waarmee een som wordt gestart.
3. Type gelijk aan 'bsub': het templatebestand bevat een bsub commando script waarmee een som kan worden toegevoegd aan de wachtrij van een rekenomgeving
4. Type gelijk aan 'general': een willekeurig ander templatebestand, dat niet voldoet aan de voorgaande definities.

In deze paragraaf wordt eerst voor het LSF en BSUB type een introductie gegeven. Daarna volgen in paragraaf 0 voorbeelden van templatebestanden aan de hand van zogenoemde user examples van veel gebruikte rekenmodellen.

2.4.1 LSF-jobscript

Een LSF-jobscript bevat code waarmee een som wordt gestart (op een rekenomgeving). Hieronder is een fictief template met een jobscript voor een som weergegeven.

LSF jobscript template

```
#!/bin/bash

#BSUB -J Q{Q:4}L{L}
#BSUB -oo Q{Q:4}L{L}.o%J
#BSUB -eo Q{Q:4}L{L}.e%J
#BSUB -n {Numcores}

DIR={ApplicationWorkingdirectory}
cd "Q{Q:4}L{L}"

echo start simulation

echo running model
model.bat -runid Q{Q:4}L{L} -bufsize 200 -back no -isddh n

echo   end simulation
```

Het script in dit voorbeeld voert de volgende handelingen uit zodra een som wordt gestart:

- Er worden een aantal BSUB-parameters gezet ('J', 'oo' en 'eo'), waarin de somID als SGWM Key wordt gebruikt en 'n' waarin de SGWM Key 'Numcores' (het aantal te gebruiken cores uit de calculationsettings) wordt gezet.
- Vervolgens wordt de (lokale) script variabele DIR gezet waarvoor de SGWM 'ApplicationWorkingdirectory' uit de calculationsettings wordt gebruikt.
- Tot slot wordt de berekening gestart door een bat bestand aan te roepen inclusief een aantal lokale variabelen.

In een jobscript template worden meestal alleen SGWM Keys van calculation-settings toegepast aangevuld met de ID van een som (als herkenning van de som). Deze kunnen gedefinieerd worden door het gebruik van SGWM Keys van parameters en variabelen. Merk nogmaals op dat alle SGWM Keys tussen accolades staan. Bovendien kan ook hier gebruik gemaakt worden van de optionele uitbreiding om het aantal karakters te definiëren en hiermee het gebruik van voorloopnullen als de lengte van een parameter of variabele korter is dan het aantal gedefinieerde karakters; zie ook paragraaf 2.2.5). SGWM zal dit template in elke map met model invoerbestanden (per som) kopiëren en de SGWM Keys vervangen. Een resultaat ziet er bijvoorbeeld zo uit:

LSF jobscript volgend uit SGWM

```
#!/bin/bash

#BSUB -J Q0500L2
#BSUB -oo Q0500L2.o%J
#BSUB -eo Q0500L2.e%J
#BSUB -n 8

DIR={ApplicationWorkingdirectory}
cd "Q0500L2"
…
echo start simulation

echo running model
model.bat -runid Q0500L2 -input -bufsize 200 -back no -isddh n

echo   end simulation
```

2.4.2 BSUB-commando script

Een template met BSUB (of soortgelijke, afhankelijk van de gebruikte rekenomgeving[21]) commando's wordt gebruikt om een som via LSF te versturen naar een rekenomgeving. De som wordt dan in een wachtrij geplaatst. Hieronder is een fictief template met een BSUB-commando script gegeven.

BSUB commando script template

```
cd Q{Q:4}L{L}
bsub -env "LSB_CONTAINER_IMAGE={ApplicationContainer}" -app
{ApplicationProfile} -q {Queue} < jobscript.bat
cd -
```

Het script in dit voorbeeld voert de volgende handelingen uit:

- Navigeer naar de map met de model invoerbestanden van een som (naam is dan gelijk aan de definitie van de somID; ook hier kan gebruik gemaakt van de optionele uitbreiding van het aantal karakters).
- Commit de som aan de wachtrij van de rekenomgeving met een BSUB-commando, waarin een aantal parameters meegegeven worden, zoals de te gebruiken Docker (ApplicationProfile), met bijbehorende modelsoftware naam (ApplicationContainer) en een prioriteitsparameter voor LSF (Queue) en tot slot het jobscript dat gestart moet worden.
- Een navigatie commando om terug te keren naar de oorspronkelijke folder.

Ook in dit template kan gebruik gemaakt worden van de verschillende SGWM Keys. Ook hier zullen dat meestal weer calculationsettings zijn.

[21] Bijvoorbeeld een QSUB-commando op de rekenomgeving van Deltares.

2.5 User Examples

In deze paragraaf worden voorbeelden gegeven van verschillende configuraties voor diverse rekenmodellen. Zowel van het PDB, het MDB als van de benodigde templatebestanden.

2.5.1 Rekenmodel Swan (Europoortgebied)

In dit voorbeeld wordt gebruik gemaakt van Swan versie 41.31A1 die als Docker (deltares/swan:41.31A.1_1.0.0) beschikbaar is gesteld op het ModellenPlatform van het NWM. Daarnaast beschouwen we alleen het Europoortgebied. De basis van het PDB wordt:

Basis Swan PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
	<Name>Swan-EUR</Name>
	<Instrumentarium>new-ve</Instrumentarium>
	<ModelSoftwarePackage>
		<Name>swan</Name>
		<Version>41.31A1</Version>
	</ModelSoftwarePackage>
	<Modelschematisation></Modelschematisation>
</PDB>
```

Er moeten sommen gemaakt worden voor 5 windsnelheden, 16 windrichtingen en 9 verschillende zeewaterstanden. We streven naar een somID die er als volgt uit ziet: 'U11D045Lp100':

- U staat voor de potentiële windsnelheid.
- D staat voor de windrichting in graden ten opzichte van Noord.
- L staat voor de zeewaterstand, waarbij een negatieve waarde aangeduid wordt beginnend met een 'n' een positieve waarde met een 'p'. Omdat we gebruik wensen te maken van een lettertoevoeging wordt de zeewaterstand in het PDB als een string gedefinieerd en niet als float.

De somID wordt dan als volgt gedefinieerd in het PDB:

SomID in PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
	<Name>Swan-EUR</Name>
	<Instrumentarium>new-ve</Instrumentarium>
	<ModelSoftwarePackage></ModelSoftwarePackage>
	<Modelschematisation>
		<Name>hr2011</Name>
		<Folder>…</Folder>
		<Geometry>
			<Folder>geometry</Folder>
		</Geometry>
		<BoundaryCondition>
			<Folder>boundary_conditions</Folder>
		</BoundaryCondition>
		<Computation>
			<Folder>computations</Folder>
			<Id>U{U:2}D{D:3}L{L}</id>
			…
		</Computation>
	</Modelschematisation>
</PDB>
```

Er worden 3 parameters in het PDB gedefinieerd.

Definitie parameters Swan PDB

```
<Parameters>
	<Parameter>
		<Name>Windsnelheid (potentieel) in m/s</Name>
		<Key>U</Key>
		<Type>float</Type>
		<Values>
			<Value>11</Value>
			<Value>22</Value>
			<Value>32</Value>
			<Value>43</Value>
			<Value>47</Value>
		</Values>
	</Parameter>
	<Parameter>
		<Name>Windrichting in graden tov Noord</Name>
		<Key>D</Key>
		<Type>float</Type>
		<Values>
			<Value>22</Value>
			<Value>45</Value>
			<Value>67</Value>
			<Value>90</Value>
			<Value>112</Value>
			<Value>135</Value>
			<Value>157</Value>
			<Value>180</Value>
			<Value>202</Value>
			<Value>225</Value>
			<Value>247</Value>
			<Value>270</Value>
			<Value>292</Value>
			<Value>315</Value>
			<Value>337</Value>
			<Value>360</Value>
		</Values>
	</Parameter>
	<Parameter>
		<Name>Waterstand in cm tov NAP</Name>
		<Key>L</Key>
		<Type>string</Type>
		<Values>
			<Value>m100</Value>
			<Value>p000</Value>
			<Value>p100</Value>
			<Value>p200</Value>
			<Value>p300</Value>
			<Value>p400</Value>
			<Value>p500</Value>
			<Value>p700</Value>
			<Value>p900</Value>
		</Values>
	</Parameter>
</Parameters>
```

Voor gebruik van de windsnelheid in het Swan model moet deze eerst vertaald worden naar een windsnelheid op open water. Hiertoe is gebruikgemaakt van onderstaande omrekentabel.

Tabel 2 Omrekentabel potentiële windsnelheid naar open water

Upotentieel (m/s)	Uopen water (m/s)
11	11.1
22	21.6
32	31.7
43	43.3
47	47.1

In het PDB definiëren we een variabele 'UU' voor de open water windsnelheid die alleen afhankelijk is van de parameter U (potentiële windsnelheid).

Definitie variabele UU Swan PDB

```
<Variables>
	<Variable>
		<Name>Windsnelheid (open water) in m/s</Name>
		<Key>UU</Key>
		<Type>float</Type>
		<Values>
			<Value>U == 11 ; 11.1</Value>
			<Value>U == 22 ; 21.6</Value>
			<Value>U == 32 ; 31.7</Value>
			<Value>U == 43 ; 43.3</Value>
			<Value>U == 47 ; 47.1</Value>
		</Values>
	</Variable>
	…
</Variables>
```

We hebben de parameter van de windrichting ('D') in het PDB zo gedefinieerd dat deze altijd met 3 karakters wordt beschreven. Zo is de eerste windrichting (gelijk aan 22.5 graden) beschreven als '022' in de somID en de laatste windrichting (gelijk aan 360 graden) beschreven als '360'.

We definiëren een extra variabele 'DD' waarin we de rekenwaarde (niet afgerond) van de windrichting opnemen. Deze is alleen afhankelijk van de parameter 'D' en ziet er als volgt uit:

Definitie variabele DD Swan PDB

```
<Variables>
	<Variable>
		<Name>Windrichting in graden tov Noord
			(rekenwaarde)</Name>
		<Key>DD</Key>
		<Type>float</Type>
		<Values>
			<Value>D == 022 ; 22.5</Value>
			<Value>D == 45 ; 45</Value>
			<Value>D == 67 ; 67.5</Value>
			<Value>D == 90 ; 90</Value>
			<Value>D == 112 ; 112.5</Value>
			<Value>D == 135 ; 135</Value>
			<Value>D == 157 ; 157.5</Value>
			<Value>D == 180 ; 180</Value>
			<Value>D == 202 ; 202.5</Value>
			<Value>D == 225 ; 225</Value>
			<Value>D == 247 ; 247.5</Value>
			<Value>D == 270 ; 270</Value>
			<Value>D == 292 ; 292.5</Value>
```

```
                    <Value>D == 315 ; 315</Value>
                    <Value>D == 337 ; 337.5</Value>
                    <Value>D == 360 ; 360</Value>
            </Values>
        </Variable>
</Variables>
```

Ook de voor de zeewaterstand moet een vertaling gemaakt worden naar een rekenwaarde. Hiertoe definiëren we de variabele 'H' die alleen afhankelijk is van de parameter 'L'. Hierin wordt ook een conversie van eenheid gedaan (van cm naar meter t.o.v. NAP).

Definitie variabele H Swan PDB

```
<Variables>
        <Variable>
                <Name>Waterstand in m tov NAP (rekenwaarde)</Name>
                <Key>H</Key>
                <Type>float</Type>
                <Values>
                        <Value>L == 'm100' ; -1.0</Value>
                        <Value>L == 'p000' ; 0.0</Value>
                        <Value>L == 'p100' ; 1.0</Value>
                        <Value>L == 'p200' ; 2.0</Value>
                        <Value>L == 'p300' ; 3.0</Value>
                        <Value>L == 'p400' ; 4.0</Value>
                        <Value>L == 'p500' ; 5.0</Value>
                        <Value>L == 'p700' ; 7.0</Value>
                        <Value>L == 'p900' ; 9.0</Value>
                </Values>
        </Variable>
</Variables>
```

Tot slot wordt in deze Swan modelschematisatie gebruik gemaakt van randvoorwaarden die afgeleid zijn met een Noordzee Swan model. In deze modelschematisatie is een afwijkende definitie van parameters gebruikt. De sommen zijn daar afhankelijk geweest van:

- de windsnelheid K in m/s [15, 20, 25, 30, 35, 40];
- de windrichting D in stappen van 30 graden [000, 210, 240, 270, 300, 330];
- de zeewaterstand L in dm t.o.v. NAP [010, 020, 030, 040, 050, 060];
- en een periodemaat P [1, 2][22];

De resultaten van deze sommen zijn beschikbaar in mappen die als volgt een naam hebben gekregen: K[windsnelheid]D[windrichting]L[waterstand]P[Periodemaat], bijvoorbeeld 'K40D330L060P2'. Om een koppeling tussen de folders (met randvoorwaarden bestanden) en onze Swan sommen te kunnen maken definiëren we 3 extra variabelen, waarin we de koppeling leggen tussen onze parameter definitie en die uit het Noordzee Swan model.

Tabel 3 Koppeling windsnelheid Swan-EUR en Noordzee Swan model

Windsnelheid Swan-EUR	Windsnelheid Noordzee
11	15
22	20
32	30
43	40
47	50

[22] (1=periodemaat -10%, 2=ongewijzigde periodemaat).

De windsnelheid wordt gekoppeld als in bovenstaande tabel met de variabele 'Urandvoorwaarden'.

Variabelen koppeling Noordzee Swan modeluitvoer voor randvoorwaarden voor windsnelheid

```
<Variables>
	<Variable>
		<Name>Windsnelheid randvoorwaarden Noordzee
			Model in m/s</Name>
		<Key>Urandvoorwaarden</Key>
		<Type>float</Type>
		<Values>
			<Value>U == 11 ; 15</Value>
			<Value>U == 22 ; 20</Value>
			<Value>U == 32 ; 30</Value>
			<Value>U == 43 ; 40</Value>
			<Value>U == 47 ; 50</Value>
		</Values>
	</Variable>
</Variables>
```

Voor de windrichting is alleen een koppeling nodig voor de westelijke windrichtingen.

Tabel 4 Koppeling windrichting Swan-EUR en Noordzee model

Windrichting Swan-EUR	Windsnelheid Noordzee
202	210
225	240
247	240
270	270
292	300
315	300
337	330
360	000

De windrichting wordt gekoppeld als in bovenstaande tabel met de variabele 'Drandvoorwaarden'.

Variabelen koppeling Noordzee Swan modeluitvoer voor randvoorwaarden voor windsnelheid

```
<Variables>
	<Variable>
		<Name>Windrichting randvoorwaarden Noordzee
			Model in graden tov Noord</Name>
		<Key>Drandvoorwaarden</Key>
		<Type>string</Type>
		<Values>
			<Value>D == 202 ; 210</Value>
			<Value>D == 225 ; 240</Value>
			<Value>D == 247 ; 240</Value>
			<Value>D == 270 ; 270</Value>
			<Value>D == 292 ; 300</Value>
			<Value>D == 315 ; 300</Value>
			<Value>D == 337 ; 330</Value>
			<Value>D == 360 ; 000</Value>
		</Values>
	</Variable>
</Variables>
```

Tabel 5 Koppeling zeewaterstand Swan-EUR en Noordzee model

Zeewaterstand Swan-EUR	Zeewaterstand Noordzee
m100	010
p000	010
p100	010
p200	020
p300	030
p400	040
p500	050
p700	060
p900	060

De zeewaterstand wordt gekoppeld als in bovenstaande tabel met de variabele 'Lrandvoorwaarden'.

Variabelen koppeling Noordzee Swan modeluitvoer voor randvoorwaarden voor windsnelheid

```
<Variables>
	<Variable>
		<Name>Waterstand randvoorwaarden Noordzee
			Model in dm tov NAP</Name>
		<Key>Lrandvoorwaarden</Key>
		<Type>float</Type>
		<Values>
			<Value>L == 'm100' ; 010</Value>
			<Value>L == 'p000' ; 010</Value>
			<Value>L == 'p100' ; 010</Value>
			<Value>L == 'p200' ; 020</Value>
			<Value>L == 'p300' ; 030</Value>
			<Value>L == 'p400' ; 040</Value>
			<Value>L == 'p500' ; 050</Value>
			<Value>L == 'p700' ; 060</Value>
			<Value>L == 'p900' ; 060</Value>
		</ParameterValues>
	</Variable>
</Variables>
```

Hiermee zijn alle variabelen en parameters voor deze user example gedefinieerd en is de definitie van het PDB compleet.

De volgende stap is het bepalen van de benodigde templates. Het primaire invoerbestand voor Swan is een SWN-bestand. Wij kiezen in deze user example om onderscheid te maken in een invoerbestand voor westelijke en voor oostelijke windrichtingen. Dit betekent dat we in het MDB twee templates van het type 'inputfile' definiëren met een conditie op de windrichting, respectievelijk 'eur_east.swn' (voor oostelijke windrichtingen van 022 tot 202) en 'eur_west.swn' (voor westelijke windrichtingen vanaf 202 t/m 360).

Template van het type 'inputfile'

```
<Templates>
	<Template>
		<Type>inputfile</Type>
		<File>eur_east.swn</File>
		<Conditions>
			<Condition>
				<Key>D</Key>
				<Value>202</Value>
				<Operator>LT</Operator>
			</Condition>
		</Conditions>
		<Update>true</Update>
	</Template>
	<Template>
		<Type>inputfile</Type>
```

```
            <File>eur_west.swn</File>
            <Conditions>
                  <Condition>
                        <Key>D</Key>
                        <Value>202</Value>
                        <Operator>GE</Operator>
                  </Condition>
            </Conditions>
            <Update>true</Update>
      </Template>
      …
</Templates>
```

Beide templatebestanden bevat SGWM Keys die bijgewerkt moeten worden. Daarom zetten we 'Update' op 'true'.

Uitgaande van het feit dat de berekeningen gedaan worden op het Modellen-Platform van SSC-Campus, maken we in elke som gebruik van een bestand 'options.sh'. In dit bestand is vastgelegd hoe de netwerk schijven/mappen 'gemount' moeten worden aan de Docker met de Swan software. Dit templatebestand bevat geen SGWM Keys en wordt niet als invoerbestand voor Swan gebruikt, noch als lsf of bsub bestand. In de definitie in het PDB wordt dit bestand dan als 'general' getypeerd en 'Update' is 'false'.

Template van het type 'general'

```
<Templates>
      …
      <Template>
            <Type>general</Type>
            <File>options.sh</File>
            <Update>false</Update>
      </Template>
</Templates>
```

Voor het uitvoeren van een Swan berekening is een lsf jobscript gedefinieerd. Dit bestand ziet er als volgt uit:

LSF Jobscript template Swan

```
#!/bin/sh
#$ -cwd
#$ -N eurU{U:2}D{D:3}L{L}

#BSUB -J eurU{U:2}D{D:3}L{L}
#BSUB -oo eurU{U:2}D{D:3}L{L}.o
#BSUB -eo eurU{U:2}D{D:3}L{L}.e
#BSUB -n {Pardocker}

swan_omp_exe=/opt/swan/swan_4131A_1_del_l64_i18_omp.exe
RUNID=eurU{U:2}D{D:3}L{L}

mkdir results

echo $PWD
echo $swan_omp_exe
echo $RUNID
echo cp $RUNID.swn INPUT
cp  $RUNID.swn INPUT

$swan_omp_exe

cp PRINT $RUNID.prt
```

De eerste regel bevat een shell commando, gevolgd door 6 regels waarin parameters van het Jobscript worden gedefinieerd. Hierin wordt gebruik gemaakt van SGWM Keys (tussen accoladen), zoals het ID van een som en de 'Pardocker', het aantal parallelle Dockers voor het uitvoeren van de Swan som. Daarna volgt de verwijzing naar de Swan executable in de Docker gevolgd door een parameter 'RUNID' waarin de somID van SGWM als SGWM Key wordt gebruikt. Er wordt een map voor de resultaten van de som aangemaakt en er worden een aantal echo commando's gegeven. Daarna wordt het Swan invoer bestand gedefinieerd als 'eur' gevolg door de SGWM somID (wederom een SGWM Key) en tot slot wordt de Swan executable gestart.

Dit Jobscript is een templatebestand dat we als volgt definiëren in het PDB:

Template van het type 'lsf' (voor het Swan jobscript)

```
<Templates>
        …
        <Template>
                <Type>lsf</Type>
                <File>lsf.job.swan4131.MPSSC</File>
                <Update>true</Update>
        </Template>
</Templates>
```

'Update' is gelijk aan 'true' omdat in het template gebruik is gemaakt van SGWM Keys.

Voor het toevoegen van sommen aan de wachtrij van LSF op het Modellen-Platform van het NWM, wordt gebruik gemaakt van bsub commando's. Deze zijn eveneens opgenomen in een templatebestand.

Dit bestand ziet er als volgt uit:

LSF bsub template Swan

```
cd U{U:2}D{D:3}L{L}
bsub -env "LSB_CONTAINER_IMAGE={Container}" -app docker -q
{Queue} < lsf.job.swan4131.MPSSC
cd ..
```

Bovengenoemde template wordt door SGWM voor elke som gekopieerd een toegevoegd aan één totaal bsub bestand. Voor elke som wordt eerst (vanuit de map waarin het bsub bestand wordt gestart; de map '…/computations') naar de map met de betreffende somID genavigeerd. In het templatebestand wordt hiervoor de somID als SGWM Key gebruikt. Daarna wordt het bsub commando van de som gedefinieerd, waarin de naam van de Docker container met Swan wordt meegegeven (ook een SGWM Key), een queue parameter voor de prioriteit van de som en de verwijzing naar het Jobscript bestand. Nadat een som via het bsub commando is toegevoegd aan de wachtrij, wordt uit de map van de som genavigeerd, terug naar de map '…/computations'.

Dit bsub-commando template definiëren we als volgt in het PDB:

Template van het type 'bsub'

```
<Templates>
        …
        <Template>
                <Type>bsub</Type>
                <File shell="#!/bin/sh">
                                bsub_file_Swan4131A1.sh</File>
                <Update>true</Update>
        </Template>
</Templates>
```

Merk op dat in de verwijzing naar het templatebestand gebruik is gemaakt van een de 'shell' optie. Hiermee zorgen we ervoor dat bij het samenstellen van het bsub bestand voor alle sommen, de eerste regel wordt gevuld met het shell commando zoals tussen de dubbel quotes is opgenomen. Als we dit zouden toevoegen aan het bsub templatebestand zou dit shell commando voor elke som herhaald worden, wat niet de bedoeling is. Ook hier is 'Update' gelijk aan 'true', aangezien er gebruik is gemaakt van SGWM Keys in het templatebestand, zoals 'ID', 'Queue' en 'Container'.

Nadat alle templatebestanden gedefinieerd zijn in het MDB, moet gedefinieerd worden of er een subfolder gebruikt wordt voor de invoerbestanden en wat de naam is van het Swan input bestand. In dit voorbeeld hebben we gekozen om geen subfolder toe te passen. De naam van het Swan invoerbestand is gelijk aan 'eurU{U}D{D}L{L}.swn'; zoals eerder ook gebruikt in het Jobscript templatebestand (zie hierboven). Dit definiëren we als volgt in het MDB:

Definitie van model input bestand

```
<Input>
        <Folder></Folder>
        <File>eurU{U:2}D{D:3}L{L}.swn</File>
</Input>
```

Als laatste onderdeel van de computation in het MDB definiëren we de zogenoemde 'states'. We onderscheiden in dit voorbeeld 4 states van elke som: 'invoer gereed', 'som gestart', 'som gereed' en 'fout opgetreden'. Voor elke state definiëren we de naam van de state, de naam van het bestand waarin naar het voorkomen van een specifieke tekst moet worden gezocht en folder waarin het bestand wordt verwacht. De definitie van de states in het MDB ziet er als volgt uit:

Definitie van states

```
<States>
        <State>
                <Name>invoer gereed</Name>
                <Text></Text>
                <File>options.sh</File>
                <Folder></Folder>
                <Progress>false</Progress>
        </State>
        <State>
                <Name>som gestart</Name>
                <Text></Text>
                <File>INPUT</File>
                <Folder></Folder>
                <Progress>false</Progress>
```

```
        </State>
        <State>
                <Name>som gereed</Name>
                <Text>Successfully completed.</Text>
                <File>*.o*</File>
                <Folder></Folder>
                <Progress>true</Progress>
        </State>
        <State>
                <Name>fout opgetreden</Name>
                <Text>Error</Text>
                <File>*.e*</File>
                <Folder></Folder>
                <Progress>false</Progress>
        </State>
</States>
```

De volgorde van de states bepaalt hoe SGWM de status van een som bepaald. In dit geval zal SGWM eerst controleren of er een bestand 'options.sh' aanwezig is in de rootmap van elke som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand 'options.sh' gezocht. Als het bestand 'options.sh' voorkomt krijgt de som de status 'invoer gereed'. Vervolgens wordt gecontroleerd of er een bestand 'INPUT' voorkomt in de rootmap van elke som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand 'INPUT' gezocht. Als het bestand 'INPUT' voorkomt krijgt de som de status 'som gestart'. Vervolgens controleert SGWM of er een bestand met '.o' in de naam voorkomt in de rootmap van elke som. Als dat zo is en daarin komt de tekst 'Successfully completed.' voor dan krijgt de som (alsnog) de status 'som gereed'. Tot slot controleert SGWM of in de rootmap van elke som of een bestand met '.e' in de naam voorkomt en of dit bestand de tekst 'Error' bevat. Als dat het geval is dan krijgt de som (alsnog) de status 'fout opgetreden'. Merk op dat het gebruik van de bestanden '.o' en '.e' een relatie heeft met de definitie van de output en error bestanden in de parameters in het Swan Jobscript.

Tot slot zijn in het MDB een drietal rekeninstellingen gedefinieerd. Allereerst een rekeninstelling voor een queue parameter in LSF. Deze kan meerdere gedefinieerde tekstwaarden aannemen, dus definiëren we deze als combo box. Omdat we de queue parameter willen kunnen aanpassen in de gebruikersschil van SGWM zetten we 'Edit' op 'true'. De tweede rekeninstelling bevat de naam van de Swan Docker Container op het ModellenPlatform bij NWM. Dit kunnen verschillende versies zijn en daarom ook als combo box gedefinieerd. Echter omdat er op dit moment slechts 1 versie beschikbaar is hebben we deze instelling niet aanpasbaar gemaakt in de gebruikersschil en dus staat 'Edit' op 'false'. De laatste rekeninstelling bevat het aantal parallel Dockers voor 1 som. Ook hier is gekozen voor een keuzelijst in een combo box en deze is te wijzigen in de gebruikersschil.

Dit ziet er dan als volgt uit in het MDB:

Definitie van rekeninstellingen in het MDB

```
<CalculationSettings>
	<CalculationSetting>
		<Name>Queue</Name>
		<Key>Queue</Key>
		<Type>combo</Type>
		<Items>RWS_normal;RWS_medium</Items>
		<Value>RWS_normal</Value>
		<Edit>true</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Docker container</Name>
		<Key>Container</Key>
		<Type>combo</Type>
		<Items>deltares/swan:41.31A.1_1.0.0</Items>
		<Value>deltares/swan:41.31A.1_1.0.0</Value>
		<Edit>false</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Docker parallel</Name>
		<Key>Pardocker</Key>
		<Type>combo</Type>
		<Items>1;2;4;8;</Items>
		<Value>4</Value>
		<Edit>true</Edit>
	</CalculationSetting>
</CalculationSettings>
```

Hiermee is de definitie van het MDB volledig.

De map met templatebestanden (‘.../computations/templates’) bevat 5 templates, waarvan het jobscript, het bsub commando script en options.sh al besproken zijn. De overige twee zijn de invoerbestanden van Swan voor respectievelijk oostelijke windrichtingen (‘eur_east.swn’) en westelijke windrichtingen (‘eur_west.swn’). In beide bestanden wordt gebruik gemaakt van diverse SGWM Keys (zowel parameters als variabelen), zoals hieronder een aantal voorbeelden (vetgedrukt) opgenomen.

Swan input-templatebestand voor westelijke windrichtingen

```
$************************U{U}D{D}L{L}**********************

$ Model                              : eur-hr2011
$ Gebied                             : Europoort
$ run id                             : eurU{U}D{D}L{L}

$ Windsnelheid U10 voor SWAN          : {UU} m/s
$ Windrichting                        : {DD} gr
$ Waterstand t.o.v. N.A.P.            : {H} m+NAP


$*****************************************************
…
$------------- algemeen -----------

SET      LEVEL={H} MAXERR=2 RHO=1025 CDCAP=0.00275 NAUTICAL
MODE     STATIONARY TWODIMENSIONAL
COORDS   CART
$------------- invoer -----------

CGRID CURV 2465 673 EXCEPT  -999   -999      CIRCLE  96
FLOW=0.015 FHIGH=1.5
READ COOR 1. '/geometry/grid_europoort_G3.grd'  IDLA=5 NHEDF=1
NHEDVEC=1 FREE
```

```
INP BOT REG 48000.0 431000.0 0  2720  2080   12.50000   12.50000
EXC -9999
READ BOT -1.0 '/geometry/europoort_bodem_2010.bod' IDLA=5 NHEDF=0

WIND  {UU} {DD} DRAG WU

INCLUDE '/geometry/eur_hr2011.fxw'

$************************************ BOUNDARY CONDITIONS
************************************

BOUN SEGM IJ 471  672 413  672   0  672 0  659 &
               0  642   0  620   0  592 0  563 &
               0  491   0  441   0  401 0  365 &
               0  275   0  139   0  106 0   83 &
               0    0 209    0 330    0        &
VAR FILE len=0.00
'/boundary_conditions/K{Urandvoorwaarden}D{Drandvoorwaarden}L{Lra
ndvoorwaarden}P2/K{Urandvoorwaarden}D{Drandvoorwaarden}L{Lrandvoo
rwaarden}P2_01.sp2' &
…
                    PAR 'results/eur{U{U}D{D}L{L}}.tst' &
                    S1D 'results/eur{U{U}D{D}L{L}}.s1d' &
                    S2D 'results/eur{U{U}D{D}L{L}}.s2d'

COMPUTE

STOP
```

2.5.2 Rekenmodel Waqua (IJssel- en Vechtdelta)

In dit voorbeeld wordt gebruik gemaakt van Waqua versie ‘simona-2016-patch-09’ die als docker (simona2016:patch09) beschikbaar is gesteld op het ModellenPlatform van het NWM. In dit voorbeeld beschouwen we de IJssel- en Vechtdelta.

De basis van het PDB wordt:

Basis Waqua PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
       <Name>WBI_WAQUA_2017</Name>
       <Instrumentarium>nwm-ve</Instrumentarium>
       <ModelSoftwarePackage>
              <Name>waqua</Name>
              <Version>simona-2016-patch-09</Version>
       </ModelSoftwarePackage>
       <Modelschematisation></Modelschematisation>
</PDB>
```

Er moeten sommen gemaakt worden voor een open en gesloten stormvloedkering, 13 afvoerniveaus op de IJssel, 6 meerpeilen, 8 windsnelheden en 7 windrichtingen. We streven naar een somID die er als volgt uit ziet: ‘KYVGQ0950Lmp090U29D247’:

- K staat voor de status van de stormvloedkering (YVG=gesloten en YVO=open overeenkomstige een falende kering)
- Q staat voor de afvoer op de IJssel in m^3/s
- L staat voor het meerpeil op het IJsselmeer t.o.v. NAP, waarbij een negatieve waarde aangeduid wordt beginnend met een ‘mn’ een positieve

waarde met een ‘mp’. Omdat we gebruik wensen te maken van een lettertoevoeging wordt het meerpeil in het PDB als een string gedefinieerd en niet als float.

- U staat voor de open water windsnelheid in m/s.
- D staat voor de windrichting in graden ten opzichte van Noord.

De somID wordt dan als volgt gedefinieerd in het PDB:

SomID in PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
	<Name>WBI_WAQUA_2017</Name>
	<Instrumentarium>new-ve</Instrumentarium>
	<ModelSoftwarePackage></ModelSoftwarePackage>
	<Modelschematisation>
		<Name>waqua-ijvd-beno14_5_hr2017-v1</Name>
		<Folder>…</Folder>
		<BoundaryCondition>
			<Folder>boundary_conditions</Folder>
		</BoundaryCondition>
		<InitialCondition>
			<Folder>initial_conditions</Folder>
		</InitialCondition>
		<Computation>
			<Folder>computations</Folder>
			<Id>K{K}Q{Q:4}L{L}U{U:2}D{D:3}</id>
			…
		</Computation>
	</Modelschematisation>
</PDB>
```

Er worden 5 parameters in het PDB gedefinieerd.

Definitie parameters Waqua PDB

```
<Parameters>
	<Parameter>
		<Name>Status stormvloedkering Ramspol</Name>
		<Key>K</Key>
		<Type>string</Type>
		<Values>
			<Value>YVG</Value>
			<Value>YVO</Value>
		</Values>
	</Parameter>
	<Parameter>
		<Name>Afvoer IJssel (Olst) in m3/s</Name>
		<Key>Q</Key>
		<Type>float</Type>
		<Values>
			<Value>100</Value>
			<Value>500</Value>
			<Value>950</Value>
			<Value>1400</Value>
			<Value>1850</Value>
			<Value>2300</Value>
			<Value>2750</Value>
			<Value>2975</Value>
			<Value>3200</Value>
			<Value>3400</Value>
			<Value>3600</Value>
			<Value>3800</Value>
			<Value>4000</Value>
		</Values>
	</Parameter>
```

```
	<Parameter>
		<Name>Meerpeil IJsselmeer in m tov NAP</Name>
		<Key>L</Key>
		<Type>string</Type>
		<Values>
			<Value>mn040</Value>
			<Value>mn010</Value>
			<Value>mp040</Value>
			<Value>mp090</Value>
			<Value>mp130</Value>
			<Value>mp150</Value>
		</Values>
	</Parameter>
	<Parameter>
		<Name>Windsnelheid (open water) in m/s</Name>
		<Key>U</Key>
		<Type>float</Type>
		<Values>
			<Value>0</Value>
			<Value>11</Value>
			<Value>18</Value>
			<Value>24</Value>
			<Value>29</Value>
			<Value>34</Value>
			<Value>39</Value>
			<Value>43</Value>
		</Values>
	</Parameter>
	<Parameter>
		<Name>Windrichting in graden tov Noord</Name>
		<Key>D</Key>
		<Type>float</Type>
		<Values>
			<Value>225</Value>
			<Value>247</Value>
			<Value>270</Value>
			<Value>292</Value>
			<Value>315</Value>
			<Value>337</Value>
			<Value>360</Value>
		</Values>
	</Parameter>
</Parameters>
```

Er worden in deze configuratie geen variabelen gebruikt. Hiermee is de definitie van het PDB gereed.

De volgende stap is het bepalen van de benodigde templates. Het primaire invoerbestand voor Waqua is een siminp bestand. Wij kiezen in deze user example om onderscheid te maken in 6 invoerbestanden respectievelijk voor 2 keringsituaties (YVO en YVG) en 3 varianten met een combinatie van een statische afvoer of een afvoergolfvorm en initieel geopende of gesloten inlaat Veesen-Wapenveld.

Dit betekent dat we in het MDB zes templates van het type ‘inputfile’ definiëren met een condities op zowel de afvoer als stormvloedkering, respectievelijk:

- ‘siminp.YVG_basis_stat_VE_geopend_VW_dicht’ bij een gesloten stormvloedkering, een statische afvoer en een initieel gesloten inlaat Veesen-Wapenveld (tot Q=1900 m^3/s);

- ‘siminp.YVG_basis_stat_VE_geopend_VW_open’ bij een gesloten stormvloedkering, een statische afvoer en een initieel geopende inlaat Veesen-Wapenveld (vanaf Q=1900 m^3/s tot en met Q=2300 m^3/s);
- ‘siminp.YVG_basis_golf_VE_geopend_VW_open’ bij een gesloten stormvloedkering, een afvoergolfvorm en een initieel geopende inlaat Veesen-Wapenveld (vanaf Q>2300 m^3/s);
- ‘siminp.YVO_basis_stat_VE_geopend_VW_dicht’ bij een open (falende) stormvloedkering, een statische afvoer en een initieel gesloten inlaat Veesen-Wapenveld (tot Q=1900 m^3/s);
- ‘siminp.YVO_basis_stat_VE_geopend_VW_open’ bij een open (falende) stormvloedkering, een statische afvoer en een initieel geopende inlaat Veesen-Wapenveld (vanaf Q=1900 m^3/s tot en met Q=2300 m^3/s);
- ‘siminp.YVO_basis_golf_VE_geopend_VW_open’ bij een open (falende) stormvloedkering, een afvoergolfvorm en een initieel geopende inlaat Veesen-Wapenveld (vanaf Q>2300 m^3/s).

Dit resulteert in de volgende templates met meervoudige condities:

Template van het type ‘inputfile’

```
<Templates>
	<Folder>computations\templates</Folder>
	<Template>
		<Type>inputfile</Type>
		<File>siminp.YVG_basis_stat_VE_
			geopend_VW_dicht</File>
		<Conditions>
			<Condition>
				<Key>Q</Key>
				<Value>1900</Value>
				<Operator>LT</Operator>
			</Condition>
			<Condition>
				<Key>K</Key>
				<Value>YVG</Value>
				<Operator>EQ</Operator>
			</Condition>
		</Conditions>
		<Update>true</Update>
	</Template>
	<Template>
		<Type>inputfile</Type>
		<File>siminp.YVG_basis_stat_VE
			_geopend_VW_open</File>
		<Conditions>
			<Condition>
				<Key>Q</Key>
				<Value>1900</Value>
				<Operator>GE</Operator>
			</Condition>
			<Condition>
				<Key>Q</Key>
				<Value>2300</Value>
				<Operator>LE</Operator>
			</Condition>
			<Condition>
				<Key>K</Key>
				<Value>YVG</Value>
				<Operator>EQ</Operator>
			</Condition>
		</Conditions>
		<Update>true</Update>
	</Template>
```

```
<Template>
	<Type>inputfile</Type>
	<File>siminp.YVG_basis_golf_VE_
		geopend_VW_open</File>
	<Conditions>
		<Condition>
			<Key>Q</Key>
			<Value>2300</Value>
			<Operator>GT</Operator>
		</Condition>
		<Condition>
			<Key>K</Key>
			<Value>YVG</Value>
			<Operator>EQ</Operator>
		</Condition>
	</Conditions>
	<Update>true</Update>
</Template>
<Template>
	<Type>inputfile</Type>
	<File>siminp.YVO_basis_stat_VE_
		geopend_VW_dicht</File>
	<Conditions>
		<Condition>
			<Key>Q</Key>
			<Value>1900</Value>
			<Operator>LT</Operator>
		</Condition>
		<Condition>
			<Key>K</Key>
			<Value>YVO</Value>
			<Operator>EQ</Operator>
		</Condition>
	</Conditions>
	<Update>true</Update>
</Template>
<Template>
	<Type>inputfile</Type>
	<File>siminp.YVO_basis_stat_VE_
		geopend_VW_open</File>
	<Conditions>
		<Condition>
			<Key>Q</Key>
			<Value>1900</Value>
			<Operator>GE</Operator>
		</Condition>
		<Condition>
			<Key>Q</Key>
			<Value>2300</Value>
			<Operator>LE</Operator>
		</Condition>
		<Condition>
			<Key>K</Key>
			<Value>YVO</Value>
			<Operator>EQ</Operator>
		</Condition>
	</Conditions>
	<Update>true</Update>
</Template>
<Template>
	<Type>inputfile</Type>
	<File>siminp.YVO_basis_golf_VE_
		geopend_VW_open</File>
	<Conditions>
		<Condition>
			<Key>Q</Key>
			<Value>2300</Value>
```

```
                        <Operator>GT</Operator>
                    </Condition>
                    <Condition>
                        <Key>K</Key>
                        <Value>YVO</Value>
                        <Operator>EQ</Operator>
                    </Condition>
                </Conditions>
                <Update>true</Update>
            </Template>
```

Alle templatebestanden bevat SGWM Keys die bijgewerkt moeten worden. Daarom zetten we 'Update' overal op 'true'.

Uitgaande van het feit dat de berekeningen gedaan worden op het Modellen-Platform van SSC-Campus, maken we in elke som gebruik van een bestand 'options.sh'. In dit bestand is vastgelegd hoe de netwerk schijven/mappen 'gemount' moeten worden aan de Docker met de Waqua software. Dit templatebestand bevat geen SGWM Keys en wordt niet als invoerbestand voor Waqua gebruikt, noch als lsf of bsub bestand. In de definitie in het MDB wordt dit bestand dan als 'general' getypeerd en 'Update' is 'false'.

Template van het type 'general'

```
<Templates>
        …
        <Template>
            <Type>general</Type>
            <File>options.sh</File>
            <Update>false</Update>
        </Template>
</Templates>
```

Voor het uitvoeren van een Waqua berekening is een lsf jobscript gedefinieerd. Dit bestand ziet er als volgt uit:

LSF Jobscript template Waqua

```
#!/bin/sh

#BSUB -J K{K}Q{Q:4}L{L}U{U:2}D{D:3}
#BSUB -oo K{K}Q{Q:4}L{L}U{U:2}D{D:3}.o%J
#BSUB -eo K{K}Q{Q:4}L{L}U{U:2}D{D:3}.e%J

# Dit is de naam van de docker container behorende bij het model
LSB_CONTAINER={ApplicationContainer}

SIMONADIR={ApplicationWorkingDirectory}
RUNID= K{K}Q{Q:4}L{L}U{U:2}D{D:3}

# profiling SIMONA-WAQUA
. $SIMONADIR/setsimonadir.sh -nodotcheck -linux64 $SIMONADIR

echo LSB_MCPU_HOSTS is: $LSB_MCPU_HOSTS
echo NUMCORES is: $LSB_DJOB_NUMPROC

env | grep LSB >> LSB_VARS.txt

cat $LSF_HOSTFILE | uniq > hostfile.gen

echo start simulation

echo running waqpre
waqpre.pl -runid $RUNID -input siminp.$RUNID -bufsize 200 -back
no -isddh n
```

```
if [ $? -ne 0 ]; then
  echo "ERROR in WAQPRE (RUNID $RUNID )"
  exit 1
fi

echo running waqpro
waqpro.pl -bufsize 200 -buf_exc 200 -buf_prt 200 -runid $RUNID -
npart $LSB_DJOB_NUMPROC -partit strip_row -isddh n -back no -
precision double

echo   end simulation
```

De eerste regel bevat een shell commando, gevolgd door 6 regels waarin parameters van het Jobscript worden gedefinieerd. Hierin wordt gebruik gemaakt van SGWM Keys (tussen accoladen), zoals het ID van een som en de werkdirectory voor het uitvoeren van de Waqua som. Daarna volg het starten van de Waqua berekening in 2 stappen. Eerst wordt 'waqpre' uitgevoerd en daarna 'waqpro'. Tot slot wordt het einde van de berekening weggeschreven.

Dit Jobscript is een templatebestand dat we als volgt definiëren in het MDB:

Template van het type 'lsf' (voor het Waqua jobscript)

```
<Templates>
        …
        <Template>
                <Type>lsf</Type>
                <File>lsf.job.script.MPSSC</File>
                <Update>true</Update>
        </Template>
</Templates>
```

'Update' is gelijk aan 'true' omdat in het template gebruik is gemaakt van SGWM Keys.

Voor het toevoegen van sommen aan de wachtrij van LSF op het Modellen-Platform van het NWM, wordt gebruik gemaakt van bsub commando's. Deze zijn eveneens opgenomen in een templatebestand. Dit bestand ziet er als volgt uit:

LSF bsub template Waqua

```
cd K{K}Q{Q:4}L{L}U{U:2}D{D:3}
bsub -app {ApplicationProfile} -q {Queue} < lsf.job.script.MPSSC
cd ..
```

Bovengenoemde template wordt door SGWM voor elke som gekopieerd een toegevoegd aan één totaal bsub bestand. Voor elke som wordt eerst (vanuit de map waarin het bsub bestand wordt gestart; de map '…/computations') naar de map met de betreffende somID genavigeerd. In het templatebestand wordt hiervoor de somID als SGWM Key gebruikt (tussen accoladen). Daarna wordt het bsub commando van de som gedefinieerd, waarin de naam van de Docker container met Waqua wordt meegegeven (ook een SGWM Key), een queue parameter voor de prioriteit van de som en de verwijzing naar het Jobscript bestand. Nadat een som via het bsub commando is toegevoegd aan de wachtrij, wordt uit de map van de som genavigeerd, terug naar de map '…/computations'.

Dit bsub-commando template definiëren we als volgt in het MDB:

Template van het type 'bsub'

```
<Templates>
        …
        <Template>
                <Type>bsub</Type>
                <File shell="#!/bin/sh">
                                bsub_file_MPSSC.sh</File>
                <Update>true</Update>
        </Template>
</Templates>
```

Merk op dat in de verwijzing naar het templatebestand gebruik is gemaakt van een de 'shell' optie. Hiermee zorgen we ervoor dat bij het samenstellen van het bsub bestand voor alle sommen, de eerste regel wordt gevuld met het shell commando zoals tussen de dubbel quotes is opgenomen. Als we dit zouden toevoegen aan het bsub templatebestand zou dit shell commando voor elke som herhaald worden, wat niet de bedoeling is. Ook hier is 'Update' gelijk aan 'true', aangezien er gebruik is gemaakt van SGWM Keys in het templatebestand, zoals 'ID', 'Queue' en 'ApplicationProfile'.

Nadat alle templatebestanden gedefinieerd zijn in het MDB, moet gedefinieerd worden hoe de map van elke som gaat heten en wat de naam is van het Waqua input bestand. In dit voorbeeld hebben we gekozen om de mapnaam van elke som gelijk te houden aan de somID. De naam van het Waqua invoerbestand wordt gelijk aan 'siminp.' Met daarachter de somID definitie; zoals eerder ook gebruikt in het Jobscript templatebestand (zie hierboven). Dit definiëren we als volgt in het MDB:

Definite van model input bestand

```
<Input>
        <Folder></Folder>
        <File>siminp.K{K}Q{Q:4}L{L}U{U:2}D{D:3}File>
</Input>
```

Als laatste onderdeel van de computation in het MDB definiëren we de zogenoemde 'states'. We onderscheiden in dit voorbeeld 4 states van elke som: 'invoer gereed', 'som gestart', 'som gereed' en 'fout opgetreden'. Voor elke state definiëren we de naam van de state, de naam van het bestand waarin naar het voorkomen van een specifieke tekst moet worden gezocht en folder waarin het bestand wordt verwacht.
De definitie van de states in het MDB ziet er als volgt uit:

Definity van states

```
<States>
        <State>
                <Name>invoer gereed</Name>
                <Text></Text>
                <File>options.sh</File>
                <Folder></Folder>
                <Progress>false</Progress>
        </State>
        <State>
                <Name>som gestart</Name>
                <Text></Text>
                <File>*.o*</File>
                <Folder></Folder>
                <Progress>false</Progress>
```

```
        </State>
        <State>
                <Name>som gereed</Name>
                <Text>end</Text>
                <File>*.o*</File>
                <Folder></Folder>
                <Progress>true</Progress>
        </State>
        <State>
                <Name>fout opgetreden</Name>
                <Text>Exited</Text>
                <File>*.e*</File>
                <Folder> </Folder>
                <Progress>false</Progress>
        </State>
</States>
```

De volgorde van de states bepaalt hoe SGWM de status van een som bepaald. In dit geval zal SGWM eerst controleren of er een bestand 'options.sh' aanwezig is in de rootmap van elke som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand 'options.sh gezocht. Als het bestand 'options.sh' voorkomt krijgt de som de status 'invoer gereed'. Vervolgens wordt gecontroleerd of er een bestand met '.o' in de naam voorkomt in de rootmap van elke som (waarin de somID als SGWM Key is gebruikt). Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand gezocht. Als een bestand met ',o' in de naam voorkomt krijgt de som de status 'som gestart'. Vervolgens controleert SGWM of in het bestand met '.o' in de naam de tekst 'end' voorkomt. Dan krijgt de som (alsnog) de status 'som gereed'. Tot slot controleert SGWM of in de rootmap van elke som een bestand met '.e' in de naam voorkomt en of dit bestand de tekst 'Exited' bevat. Als dat het geval is dan krijgt de som (alsnog) de status 'fout opgetreden'. Merk op dat het gebruik van de bestanden '.o' en '.e' een relatie heeft met de definitie van de output en error bestanden in de parameters in het Waqua Jobscript.

Tot slot zijn in het MDB een 5 rekeninstellingen gedefinieerd. Allereerst een rekeninstelling voor een queue parameter in LSF. Deze kan meerdere gedefinieerde tekstwaarden aannemen, dus definiëren we deze als combo box. Omdat we de queue parameter willen kunnen aanpassen in de gebruikersschil van SGWM zetten we 'Edit' op 'true'. De tweede instelling geeft aan dat we gebruik maken van een docker en deze hoeft niet aangepast te worden in de gebruikersschil en daarom is 'Edit' gelijk aan 'false'. De derde rekeninstelling bevat de naam van de Waqua Container op het ModellenPlatform bij NWM. Dit kunnen verschillende versies zijn en daarom ook als combo box gedefinieerd. Echter omdat er op dit moment slechts 1 versie beschikbaar is hebben we deze instelling niet aanpasbaar gemaakt in de gebruikersschil en dus staat 'Edit' op 'false'. In het jobscript wordt gebruik gemaakt van een workingdirectory. Ook hiervoor is een rekeninstelling gemaakt, maar die is niet te wijzigen in de gebruikersschil. De laatste rekeninstelling bevat een specifiek parameter instelling voor Waqua, namelijk de initiële tijd. Deze is door de gebruiker te wijzigen in de gebruikersschil.

Dit ziet er dan als volgt uit in het MDB:

Definitie van rekeninstellingen in het PDB

```
<CalculationSettings>
	<CalculationSetting>
		<Name>Queue</Name>
		<Key>Queue</Key>
		<Type>combo</Type>
		<Items>RWS-normal;RWS_medium;RWS_large</Items>
		<Value>RWS-normal</Value>
		<Edit>true</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Application Profile</Name>
		<Key>ApplicationProfile</Key>
		<Type>string</Type>
		<Value>docker</Value>
		<Edit>false</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Application Container</Name>
		<Key>ApplicationContainer</Key>
		<Type>combo</Type>
			<Items>simona2016:patch09</Items>
		<Value>simona2016:patch09</Value>
		<Edit>false</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Application Working Directory</Name>
		<Key>ApplicationWorkingDirectory</Key>
		<Type>string</Type>
		<Value>mnt/SGWM</Value>
		<Edit>false</Edit>
	</CalculationSetting>
		<CalculationSetting>
		<Name>Inital time</Name>
		<Key>IT</Key>
		<Type>float</Type>
		<Value>14400</Value>
		<Edit>true</Edit>
	</CalculationSetting>
</CalculationSettings>
```

Hiermee is de definitie van het MDB volledig.

De map met templatebestanden (‘.../computations/templates’) bevat 11 templates, waarvan het jobscript, het bsub commando script en options.sh al besproken zijn. De overige zes zijn de invoerbestanden van Waqua. In deze bestanden wordt gebruik gemaakt van diverse SGWM Keys (zowel parameters, variabelen als rekeninstellingen), zoals hieronder een aantal voorbeelden (vetgedrukt) opgenomen.

Waqua input-templatebestand

```
# siminp template voor model waqua-ijvd-beno14_5_hr2017-v1

# Aanpassingen tov basisschematisatie:
# - Alle siminp-verwijzingen naar Bypass fase 1b verwijderd
# - WTI randvoorwaarden overgenomen (zie onder)
# - Uitvoerpunten Reevediep en hmpunten IJssel toegevoegd
# - SDSOUTPUT, MAPS, WIND uitgezet
# - TIROUC= 0.25 ipv 5.

set noecho
```

```
#
================================================================
====
# identification experiment
#
================================================================
====
# Simulatie t.b.v. Optimalisatie inlaatdrempel Reevediep
#    Voorliggende simulatie gaat uit van
#       meerpeil {L} (mn = negatief, mp = positief)
#       afvoer IJssel {Q} (afvoer Vecht is hieraan gekoppeld)
#       open-water windsnelheid {U} (in m/s)
#       windrichting {D} (360 = noord, 180 = zuid)
#    Initiele condities bij Q{Q:4}L{L},
#       tijdstip voor inlezen is {IT}
#    Initiele positie Veesen-Wapenveld is open
#    Initiele positie stuwen Vecht is open
#    Op de benedenrand bij Kornwerderzand en Den Oever is een
afvoerontrekking
#       opgelegd die is afgeleid uit een Q-h simulatie met
hetzelfde meerpeil
#       en dezelfde afvoer op de IJssel.
#
# Voorliggende simulatie is K{K}Q{Q:4}L{L}U{U:2}D{D:3}

IDENTIFICATION

  WAQUA
    EXPERIMENT = 'K{K}Q{Q:4}L{L}U{U:2}D{D:3}' OVERWRITE
    MODID      = 'K{K}Q{Q:4}L{L}U{U:2}D{D:3}'
    TITLE      = 'K{K}Q{Q:4}L{L}U{U:2}D{D:3}'

#
================================================================
====
…
#
================================================================
====
# general model constants
#
================================================================
====

GENERAL
  PHYSICAL_PARAMETERS
    GRAVITY    =    9.8130
    WATDENSITY = 1000.0
    AIRDENSITY =    1.2265

  WIND
   WCONVERSIONFACTOR = 1.0
   WUNIT = 'm/s'
   VARIABLE_CD
    CDA = 1.36673E-03
    CDB = 3.90E-3
    WIND_CDA = 7.77886
    WIND_CDB = 50.0
    INCLUDE =
'../../../boundary_conditions/wind/rvw_wind_const_afv_D{D}_U{U}.t
xt'
…
#     initiele condities uit inspeelsimulatie Q{Q:4}L{L}
    INITIAL
       READ_FROM
#        EXP_INITIAL='Q{Q:4}L{L}'
```

```
        SDS_INITIAL='../../../initial_conditions/Q{Q:4}L{L}/SDS-
Q{Q:4}L{L}'
        TIME_INITIAL={IT}

    BOUNDARIES
      B: open1, BTYPE='disch-ad', BDEF='Series ',REFL= 0.0 SAME
# Olst
      B: open2, BTYPE='disch-ad', BDEF='Series ',REFL= 0.0 SAME
# Ommen
      B: open3, BTYPE='disch-ad', BDEF='Series ',REFL= 0.0 SAME
# Den Oever
      B: open4, BTYPE='disch-ad', BDEF='Series ',REFL= 0.0 SAME
# Kornwerderzand

    TIMESERIES
      INCLUDE
'../../../boundary_conditions/flow/L{L}_Q{Q:4}/Q_Olst.001'
      INCLUDE
'../../../boundary_conditions/flow/L{L}_Q{Q:4}/Q_Ommen.001'
      INCLUDE
'../../../boundary_conditions/flow/L{L}_Q{Q:4}/Q_Den_Oever.001'
      INCLUDE
'../../../boundary_conditions/flow/L{L}_Q{Q:4}/Q_Kornwerderzand.0
01'

    DISCHARGES
      INCLUDE
'../../../boundary_conditions/flow/L{L}_Q{Q:4}/Q_lateraal.001'

    BARRIERS
      INCLUDE '../../../geometry/kunstwerken/stuw-barrier-
sturing_VE_geopend_VW_open-v2.001'
…
```

2.5.3 Rekenmodel D-Hydro (Maas en Rijn)

Maas

In dit eerste voorbeeld wordt gebruik gemaakt van D-Hydro versie 2020-03 die als docker (2.10.10_65673_delft3dfm_2020_03) beschikbaar is gesteld op het ModellenPlatform van het NWM. In eerste instantie beschouwen we het gebied van de Maas.

De basis van het PDB wordt:

Basis D-Hydro PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
       <Name>D-Hydro_Maas</Name>
       <Instrumentarium>nwm-ve</Instrumentarium>
       <ModelSoftwarePackage>
              <Name>d-hydro</Name>
              <Version>2020-03</Version>
       </ModelSoftwarePackage>
       <Modelschematisation></Modelschematisation>
</PDB>
```

Er moeten sommen gemaakt worden voor 3 afvoeren. We streven naar een somID die er als volgt uit ziet: ‘Maas_Q50’ waarin Q staat voor de afvoer op de Maas in m^3/s.

De somID wordt dan als volgt gedefinieerd in het PDB:

SomID in PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
	<Name>D-Hydro_Maas</Name>
	<Instrumentarium>new-ve</Instrumentarium>
	<ModelSoftwarePackage></ModelSoftwarePackage>
	<Modelschematisation>
		<Name>BOI</Name>
		<Folder>…</Folder>
		<Geometry>
			<Folder>geometry</Folder>
		</Geometry>
		<BoundaryCondition>
			<Folder>boundary_conditions</Folder>
		</BoundaryCondition>
		<InitialCondition>
			<Folder>initial_conditions</Folder>
		</InitialCondition>
		<Computation>
			<Folder>computations</Folder>
			<Id>maasQ{Q:4}</id>
			…
		</Computation>
	</Modelschematisation>
</PDB>
```

De afvoer wordt één parameter in het PDB.

definite parameters D-Hydro PDB

```
<Parameters>
	<Parameter>
		<Name>Afvoer op de Maas in m3/s</Name>
		<Key>Q</Key>
		<Type>float</Type>
		<Values>
			<Value>50</Value>
			<Value>250</Value>
			<Value>1500</Value>
		</Values>
	</Parameter>
</Parameters>
```

Er worden in deze configuratie geen variabelen gebruikt. Hiermee is de definitie van het PDB gereed.

De volgende stap is het bepalen van de benodigde templates. Het primaire invoerbestand voor D-Hydro is een MDU-bestand. We onderscheiden in deze user example één invoerbestand voor D-Hydro.

Template van het type 'inputfile'

```
<Templates>
	<Template>
		<Type>inputfile</Type>
		<File>dflowfm2d-maas-beno19_6-w4.mdu</File>
		<Update>true</Update>
	</Template>
	…
</Templates>
```

Dit templatebestand bevat SGWM Keys die bijgewerkt moeten worden. Daarom zetten we 'Update' op 'true'.

In D-Hydro wordt gebruik gemaakt van een DIMR-configuratie. We onderscheiden in deze user example 4 verschillende DIMR-configuraties. Een configuratie met en zonder gebruik van RTC en een configuratie waarbij parallel wordt gerekend en serieel. Voor elke configuratie variant hebben we een template opgesteld:

- 'dimr_config_SER.xml': DIMR zonder RTC-module en serieel rekenen;
- 'dimr_config_PAR.xml': DIMR zonder RTC-module en parallel rekenen;
- 'dimr_config_PAR_RTC.xml': DIMR met RTC-module en serieel rekenen;
- 'dimr_config_SER_RTC.xml': DIMR met RTC-module en parallel rekenen.

We hebben ervoor gekozen om de keuze voor serieel/parallel rekenen en ook met of zonder RTC door de gebruiker te kunnen laten maken bij het aanmaken van een sommenset in SGWM. We hebben hiervoor SGWM-rekeninstellingen gedefinieerd, respectievelijk 'CalcType' en 'SturingType' (zie verderop). Voor de verschillende DIMR-templates hebben we condities gedefinieerd die afhankelijk zijn van de SGWM-rekeninstellingen. Om dit kenbaar te maken aan SGWM voegen we het attribuut 'calculationsetting= "true"' toe aan de condities van de templates.

Templates van het type 'general' voor de DIMR-configuratie

```
<Templates>
	<Template>
		<Type>general</Type>
		<File>dimr_config_SER.xml</File>
		<Conditions>
			<Condition calculationsetting="true">
				<Key>CalcType</Key>
				<Value>SER</Value>
				<Operator>EQ</Operator>
			</Condition>
			<Condition calculationsetting="true">
				<Key>SturingType</Key>
				<Value>zonderRTC</Value>
				<Operator>EQ</Operator>
			</Condition>
		</Conditions>
		<Update>false</Update>
	</Template>
	<Template>
		<Type>general</Type>
		<File>dimr_config_PAR.xml</File>
		<Conditions>
			<Condition calculationsetting="true">
				<Key>CalcType</Key>
				<Value>PAR</Value>
				<Operator>EQ</Operator>
			</Condition>
			<Condition calculationsetting="true">
				<Key>SturingType</Key>
				<Value>zonderRTC</Value>
				<Operator>EQ</Operator>
			</Condition>
		</Conditions>
		<Update>false</Update>
	</Template>
	<Template>
		<Type>general</Type>
		<File>dimr_config_PAR_RTC.xml</File>
		<Conditions>
			<Condition calculationsetting="true">
				<Key>CalcType</Key>
```

```
                        <Value>PAR</Value>
                        <Operator>EQ</Operator>
                    </Condition>
                    <Condition calculationsetting="true">
                        <Key>SturingType</Key>
                        <Value>metRTC</Value>
                        <Operator>EQ</Operator>
                    </Condition>
                </Conditions>
                <Update>false</Update>
            </Template>
            <Template>
                <Type>general</Type>
                <File>dimr_config_SER_RTC.xml</File>
                <Conditions>
                    <Condition calculationsetting="true">
                        <Key>CalcType</Key>
                        <Value>SER</Value>
                        <Operator>EQ</Operator>
                    </Condition>
                    <Condition calculationsetting="true">
                        <Key>SturingType</Key>
                        <Value>metRTC</Value>
                        <Operator>EQ</Operator>
                    </Condition>
                </Conditions>
                <Update>false</Update>
            </Template>
            …
</Templates>
```

De DIMR-templatebestanden zijn geen inputfile van D-Hydro en vallen daarom onder het type ‘general’. Er zijn ook geen SGWM Keys opgenomen in deze templatebestanden, dus is ‘Edit’ gelijk aan ‘false’.

De D-Hydro som maakt verder gebruik van een netCDF bestand en een bestand met randvoorwaarden en laterale toestromingen. Deze zijn gedefinieerd per afvoer niveau. Voor elke afvoer is een aparte template aangemaakt, dus maken we gebruik van condities. Bovengenoemde templates zijn allemaal van het type ‘general’ en bevatten geen SGWM Keys.

Overige templates van het type ‘general’

```
<Templates>
            …
            <Template>
                <Type>general</Type>
                <File>dflowfm2d-maas-beno19_6-w4_net.nc</File>
                <Update>false</Update>
            </Template>
            <Template>
                <Type>general</Type>
                <File>Maas_S50_bnd.ext</File>
                <Conditions>
                    <Condition>
                        <Key>Q</Key>
                        <Value>50</Value>
                        <Operator>EQ</Operator>
                    </Condition>
                </Conditions>
                <Update>false</Update>
            </Template>
            <Template>
                <Type>general</Type>
                <File>Maas_S250_bnd.ext</File>
```

```
            <Conditions>
                  <Condition>
                        <Key>Q</Key>
                        <Value>250</Value>
                        <Operator>EQ</Operator>
                  </Condition>
            </Conditions>
            <Update>false</Update>
      </Template>
      <Template>
            <Type>general</Type>
            <File>Maas_S1500_bnd.ext</File>
            <Conditions>
                  <Condition>
                        <Key>Q</Key>
                        <Value>1500</Value>
                        <Operator>EQ</Operator>
                  </Condition>
            </Conditions>
            <Update>false</Update>
      </Template>
      …
</Templates>
```

Uitgaande van het feit dat de berekeningen gedaan worden op het Modellen-Platform van SSC-Campus, maken we in elke som gebruik van een bestand 'options.sh'. In dit bestand is vastgelegd hoe de netwerk schijven/mappen 'gemount' moeten worden aan de docker met de D-Hydro software. Dit templatebestand bevat geen SGWM Keys en wordt niet als invoerbestand voor D-Hydro gebruikt, noch als lsf of bsub bestand. In de definitie in het PDB wordt dit bestand dan als 'general' getypeerd en 'Update' is 'false'.

Template van het type 'general'

```
<Templates>
      …
      <Template>
            <Type>general</Type>
            <File>options.sh</File>
            <Update>false</Update>
      </Template>
</Templates>
```

Voor het uitvoeren van een D-Hydro berekening is een lsf jobscript gedefinieerd. Dit bestand ziet er als volgt uit:

LSF Jobscript template D-Hydro

```
#!/bin/bash
#$ -cwd

#BSUB -J maas{CalcType}
#BSUB -oo log{CalcType}.o
#BSUB -eo log{CalcType}.e

cd maas_Q{Q}

if [ '{CalcType}' == 'PAR' ]
then
      echo "Run parallel preprocessing"
      export LD_LIBRARY_PATH=/opt/delft3dfm_latest/lnx64/lib:
            $LD_LIBRARY_PATH
      /opt/delft3dfm_latest/lnx64/bin/dflowfm --partition:
            ndomains=$(($(echo "'{PARDocker}'" | grep -o " " |
            wc -l)+1)):icgsolver=6 dflowfm2d-maas-beno19_6-
            w4.mdu
```

```
        # Wait for all jobs to complete
        wait
fi

ulimit -s unlimited
if [ '{CalcType}' == 'PAR' ]
then
        echo "Run parallel"
        if [ '{SturingType}' == 'metRTC' ]
        then
                /opt/delft3dfm_latest/lnx64/bin/run_dimr.sh
                        --debug 9 --dockerparallel -c $(($(echo
                        "'{PARDocker}'" | grep -o " " | wc -l)+1))
                        --D3D_HOME /opt/delft3dfm_latest/lnx64
                        -m dimr_config_PAR_RTC.xml
        else
                /opt/delft3dfm_latest/lnx64/bin/run_dimr.sh
                        --debug 9 --dockerparallel -c $(($(echo
                        "'{PARDocker}'" | grep -o " " | wc -l)+1))
                        --D3D_HOME /opt/delft3dfm_latest/lnx64
                        -m dimr_config_PAR.xml
        fi
else
        echo "Run serial"
        if [ '{SturingType}' == 'metRTC' ]
        then
                /opt/delft3dfm_latest/lnx64/bin/run_dimr.sh
                        --debug 9 --dockerparallel -c 1
                        --D3D_HOME /opt/delft3dfm_latest/lnx64
                        -m dimr_config_SER_RTC.xml
        else
                /opt/delft3dfm_latest/lnx64/bin/run_dimr.sh
                        --debug 9 --dockerparallel -c 1
                        --D3D_HOME /opt/delft3dfm_latest/lnx64
                        -m dimr_config_SER.xml
        fi
fi
```

De eerste regel bevat een shell commando, gevolgd door 3 regels waarin parameters van het Jobscript worden gedefinieerd. Hierin wordt gebruik gemaakt van SGWM Keys (tussen accoladen), zoals het ID van een som en de 'CalcType'. Wordt gekozen om parallel te rekenen dan moet er eerst een preprocessing stap uitgevoerd worden waarbij de D-Hydro modelschematisatie wordt opgeknipt, zodat er parallel gerekend kan worden. Als die stap is uitgevoerd dan volgt eveneens afhankelijk van de keuze voor serieel of parallel rekenen een commando om D-Hydro te starten. Hierbij wordt zowel voor serieel of parallel rekenen eerst ook nog een onderscheid gemaakt in reken met of zonder RTC-modules. Afhankelijk van de keuze wordt verwezen naar het overeenkomstige DIMR configuratie-bestand.

Dit Jobscript is een templatebestand dat we als volgt definiëren in het MDB:

Template van het type 'lsf' (voor het D-Hydro jobscript)

```
<Templates>
        …
        <Template>
                <Type>lsf</Type>
                <File>lsf.job.script.MPSSC</File>
                <Update>true</Update>
        </Template>
</Templates>
```

'Update' is gelijk aan 'true' omdat in het template gebruik is gemaakt van SGWM Keys.

Voor het toevoegen van sommen aan de wachtrij van LSF op het Modellen-Platform van het NWM, wordt gebruik gemaakt van bsub commando's. Deze zijn eveneens opgenomen in een templatebestand. Dit bestand ziet er als volgt uit:

LSF bsub template D-Hydro

```
cd maas_Q{Q:4}/input
bsub -env "LSB_CONTAINER_IMAGE={ApplicationContainer}" -app
docker -q {Queue} < lsf.job.script.MPSSC
cd ..
cd ..
```

Bovengenoemde template wordt door SGWM voor elke som gekopieerd een toegevoegd aan één totaal bsub bestand. Voor elke som wordt eerst (vanuit de map waarin het bsub bestand wordt gestart; de map '…/computations') naar de map met de betreffende somID genavigeerd. In het templatebestand wordt hiervoor de somID als SGWM Key gebruikt (tussen accoladen). Daarna wordt het bsub commando van de som gedefinieerd, waarin de naam van de Docker container met D-Hydro wordt meegegeven (ook een SGWM Key), een queue parameter voor de prioriteit van de som en de verwijzing naar het Jobscript bestand. Nadat een som via het bsub commando is toegevoegd aan de wachtrij, wordt uit de map van de som genavigeerd, terug naar de map '…/computations'.

Dit bsub-commando template definiëren we als volgt in het MDB:

Template van het type 'bsub'

```
<Templates>
        …
        <Template>
                <Type>bsub</Type>
                <File shell="#!/bin/bash">bsub_file_MPSSC.sh</File>
                <Update>true</Update>
        </Template>
</Templates>
```

Merk op dat in de verwijzing naar het templatebestand gebruik is gemaakt van een de 'shell' optie. Hiermee zorgen we ervoor dat bij het samenstellen van het bsub bestand voor alle sommen, de eerste regel wordt gevuld met het shell commando zoals tussen de dubbel quotes is opgenomen. Als we dit zouden toevoegen aan het bsub templatebestand zou dit shell commando voor elke som herhaald worden, wat niet de bedoeling is. Ook hier is 'Update' gelijk aan 'true', aangezien er gebruik is gemaakt van SGWM Keys in het templatebestand, zoals 'ID', 'Queue' en 'ApplicationContainer'.

Nadat alle templatebestanden gedefinieerd zijn in het MDB, moet gedefinieerd worden of er een subfolder gebruikt wordt voor de invoerbestanden en wat de naam is van het D-Hydro input bestand. In dit voorbeeld hebben we gekozen om de subfolder 'input' aan te laten maken. De naam van het D-Hydro invoerbestand is gelijk aan de originele naam van het template.

Dit definiëren we als volgt in het MDB:

Definitie van model input bestand

```
<Input>
       <Folder>input</Folder>
       <File>dflowfm2d-maas-beno19_6-w4.mdu</File>
</Input>
```

Als laatste onderdeel van de modelschematisatie in het MDB definiëren we de zogenoemde 'states'. We onderscheiden in dit voorbeeld 4 states van elke som: 'invoer gereed', 'som gestart', 'som gereed' en 'fout opgetreden'. Voor elke state definiëren we de naam van de state, de naam van het bestand waarin naar het voorkomen van een specifieke tekst moet worden gezocht en folder waarin het bestand wordt verwacht. De definitie van de states in het PDB ziet er als volgt uit:

Definitie van states

```
<States>
       <State>
              <Name>invoer gereed</Name>
              <Text></Text>
              <File>*.mdu</File>
              <Folder>input</Folder>
              <Progress>false</Progress>
       </State>
       <State>
              <Name>som gestart</Name>
              <Text></Text>
              <File>*.dia</File>
              <Folder>results</Folder>
              <Progress>false</Progress>
       </State>
       <State>
              <Name>som gereed</Name>
              <Text>Successfully completed.</Text>
              <File>*.o.*</File>
              <Folder></Folder>
              <Progress>false</Progress>
       </State>
       <State>
              <Name>fout opgetreden</Name>
              <Text>exit code</Text>
              <File>*.o*</File>
              <Folder> </Folder>
              <Progress>false</Progress>
       </State>
</States>
```

De volgorde van de states bepaalt hoe SGWM de status van een som bepaald. In dit geval zal SGWM eerst controleren of er een bestand met de extensie 'mdu' aanwezig is in de subfolder 'input' van elke som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand gezocht. Als het bestand met de extensie 'mdu' voorkomt krijgt de som de status 'invoer gereed'. Vervolgens wordt gecontroleerd of er een bestand met een extensie 'dia' voorkomt in de map 'results' binnen een map van de betreffende som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand gezocht. Als het bestand met de extensie 'dia' voorkomt krijgt de som de status 'som gestart'. Vervolgens controleert SGWM of in de rootmap van elke som een bestand met '.o' in de naam voorkomt en of in dit

bestand de tekst ‘Successfully completed.’ voorkomt. Als dat zo dan krijgt de som (alsnog) de status ‘som gereed’. Tot slot controleert SGWM of in de rootmap van elke som een bestand met ‘.e’ in de naam voorkomt en of dit bestand de tekst ‘Error’ bevat. Als dat het geval is dan krijgt de som (alsnog) de status ‘fout opgetreden’. Merk op dat het gebruik van de bestanden ‘.o’ en ‘.e’ een relatie heeft met de definitie van de output en error bestanden in de parameters in het D-Hydro Jobscript.

Tot slot zijn in het MDB zes rekeninstellingen gedefinieerd. Allereerst een rekeninstelling voor een queue parameter in LSF. Deze kan meerdere gedefinieerde tekstwaarden aannemen, dus definiëren we deze als combo box. Omdat we de queue parameter willen kunnen aanpassen in de gebruikersschil van SGWM zetten we ‘Edit’ op ’true’. De tweede en derde rekeninstelling bevat de informatie voor de D-Hydro Docker Container op het ModellenPlatform bij NWM. Dit kunnen verschillende versies zijn en daarom ook als combo box gedefinieerd. Echter omdat er op dit moment slechts 1 versie beschikbaar is hebben we deze instelling niet aanpasbaar gemaakt in de gebruikersschil en dus staat ‘Edit’ op ‘false’. De vierde en vijfde rekeninstelling zijn er om de gebruiker te laten kiezen tussen een seriële of parallelle berekening met respectievelijk zonder RTC-module. Omdat de gebruiker dit moet kunnen kiezen staat ‘Edit’ op ’true’. De laatste rekeninstelling heeft te maken met het aantal cores in een parallelle berekening en is niet te aan te passen voor de gebruiker. Dit ziet er dan als volgt uit in het MDB:

Definitie van rekeninstellingen in het PDB

```
<CalculationSettings>
	<CalculationSetting>
		<Name>Queue</Name>
		<Key>Queue</Key>
		<Type>combo</Type>
		<Items>RWS_normal;RWS_medium;RWS_large</Items>
		<Value>RWS_medium</Value>
		<Edit>true</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Application Profile</Name>
		<Key>ApplicationProfile</Key>
		<Type>combo</Type>
		<Items>docker</Items>
		<Value>docker</Value>
		<Edit>false</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Application Container</Name>
		<Key>ApplicationContainer</Key>
		<Type>combo</Type>
		<Items>deltares/delft3dfm:2021.04_1.0.0</Items>
		<Value>deltares/delft3dfm:2021.04_1.0.0</Value>
		<Edit>false</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Type som</Name>
		<Key>CalcType</Key>
		<Type>combo</Type>
		<Items>SER;PAR</Items>
		<Value>PAR</Value>
		<Edit>true</Edit>
	</CalculationSetting>
```

```
        <CalculationSetting>
                <Name>Type sturing</Name>
                <Key>SturingType</Key>
                <Type>combo</Type>
                <Items>zonderRTC;metRTC</Items>
                <Value>zonderRTC</Value>
                <Edit>true</Edit>
        </CalculationSetting>
        <CalculationSetting>
                <Name>Docker parallel</Name>
                <Key>PARDocker</Key>
                <Type>combo</Type>
                <Items>0 1 2 3 4 5 6 7;</Items>
                <Value>0 1 2 3 4 5 6 7</Value>
                <Edit>false</Edit>
        </CalculationSetting>
</CalculationSettings>
```

Hiermee is de definitie van het MDB volledig.

De map met templatebestanden ('.../computations/templates') bevat 12 templates, waarvan het jobscript, het bsub commando script, options.sh en de 4 DIMR config-bestanden al besproken zijn. Het MDU-bestand is het invoerbestand van D-Hydro en bevat 1 SGWM Key, een verwijzing naar een bestand met randvoorwaarden. Omdat het bestand verder zeer uitgebreid is wordt dit bestand hier niet weergegeven. De overige bestanden (de bestanden met randvoorwaarden en een NetCDF bestand) worden alleen naar elke som map gekopieerd.

Rijn

In het tweede voorbeeld wordt gebruik gemaakt van D-Hydro versie 2022-01 die als docker (deltares/delft3dfm:2022.01patch01_1.0.0) beschikbaar is gesteld op het ModellenPlatform van het NWM. We beschouwen nu het gebied van de Rijn.

De basis van het PDB wordt:

Basis D-Hydro PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
        <Name>DHYDRO-rijn</Name>
        <Instrumentarium>nwm-ve</Instrumentarium>
        <ModelSoftwarePackage>
                <Name>d-hydro</Name>
                <Version> 2022-01patch01</Version>
        </ModelSoftwarePackage>
        <Modelschematisation></Modelschematisation>
</PDB>
```

Er moeten sommen gemaakt worden voor 15 verschillende afvoeren. We streven naar een somID die afhankelijk is van de afvoer op de Rijn in m^3/s en een indicator die aangeeft of er sprake is van een stationaire ('S') of dynamische afvoerberekening ('D').

De somID wordt dan als volgt gedefinieerd in het PDB:

SomID in PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
	<Name>D-Hydro_Maas</Name>
	<Instrumentarium>new-ve</Instrumentarium>
	<ModelSoftwarePackage></ModelSoftwarePackage>
	<Modelschematisation>
		<Name>hr2023</Name>
		<Folder>…</Folder>
		<Geometry>
			<Folder>geometry</Folder>
		</Geometry>
		<BoundaryCondition>
			<Folder>boundary_conditions</Folder>
		</BoundaryCondition>
		<InitialCondition>
			<Folder>initial_conditions</Folder>
		</InitialCondition>
		<Computation>
			<Folder>computations\hr\hr2023</Folder>
			<Id>rijn_Q{TypeAfvoer}{Q:5}</id>
			…
		</Computation>
	</Modelschematisation>
</PDB>
```

De afvoer wordt één parameter in het PDB.

definite parameters D-Hydro PDB

```
<Parameters>
	<Parameter>
		<Name>Afvoer op de Rijn in m3/s</Name>
		<Key>Q</Key>
		<Type>float</Type>
		<Values>
			<Value>600</Value>
			<Value>2000</Value>
			<Value>4000</Value>
			<Value>6000</Value>
			<Value>8000</Value>
			<Value>10000</Value>
			<Value>12000</Value>
			<Value>13000</Value>
			<Value>14000</Value>
			<Value>15000</Value>
			<Value>16000</Value>
			<Value>17000</Value>
			<Value>18000</Value>
			<Value>20000</Value>
			<Value>24000</Value>
		</Values>
	</Parameter>
</Parameters>
```

Afhankelijk van de afvoer moeten een aantal aanvullende waarden bepaald kunnen worden. Deze koppelingen maken we aan via de definitie van variabelen in het PDB. In eerste instantie beschouwen we berekeningen met een stationaire (tot en met 4000 m^3/s) en dynamische afvoer (vanaf 4000 m^3/s). Merk op dat deze variabele ook gebruikt wordt in de somID. Daarnaast worden ook nog een tweetal stoptijdstippen (‘C time’ en ‘RTC tstop’) afgeleid op basis van de afvoer, een RTC werkmap (‘RTC workingdir’) en een Ini bestand aanduiding (‘Ini Field’).

Definitie variabelen SOBEK in PDB

```
<Variables>
	<Variable>
		<Name>Type afvoer </Name>
		<Key>TypeAfvoer</Key>
		<Type>string</Type>
		<Values>
			<Value>Q &lt;= 4000 ; S</Value>
			<Value>Q &gt; 4000 ; D</Value>
		</Value>
	</Variable>
	<Variable>
		<Name>C time</Name>
		<Key>TStop</Key>
		<Type>float</Type>
		<Values>
			<Value>Q &lt;= 4000 ; 21600</Value>
			<Value>Q &gt; 4000 ; 25920</Value>
		</Values>
	</Variable>
	<Variable>
		<Name>RTC tstop</Name>
		<Key>RTC_T</Key>
		<Type>string</Type>
		<Values>
			<Value>Q &lt;= 4000 ; 129600</Value>
			<Value>Q &gt; 4000 ; 1555200</Value>
		</Values>
	</Variable>
	<Variable>
		<Name>RTC workingdir</Name>
		<Key>RTC_dir</Key>
		<Type>string</Type>
		<Values>
			<Value>Q == 600 ; rtc_ini1</Value>
			<Value>Q == 2000 ; rtc_ini1</Value>
			<Value>Q == 4000 ; rtc_ini2</Value>
			<Value>Q == 6000 ; rtc_ini1</Value>
			<Value>Q == 8000 ; rtc_ini1</Value>
			<Value>Q == 10000 ; rtc_ini1</Value>
			<Value>Q == 12000 ; rtc_ini1</Value>
			<Value>Q == 13000 ; rtc_ini1</Value>
			<Value>Q == 14000 ; rtc_ini1</Value>
			<Value>Q == 15000 ; rtc_ini1</Value>
			<Value>Q == 16000 ; rtc_ini1</Value>
			<Value>Q == 17000 ; rtc_ini2</Value>
			<Value>Q == 18000 ; rtc_ini2</Value>
			<Value>Q == 20000 ; rtc_ini2</Value>
			<Value>Q == 24000 ; rtc_ini2</Value>
		</Values>
	</Variable>
	<Variable>
		<Name>Ini Field</Name>
		<Key>Ini</Key>
		<Type>string</Type>
		<Values>
			<Value>Q == 600 ; S__600</Value>
			<Value>Q == 2000 ; S_2000</Value>
			<Value>Q == 4000 ; S_4000</Value>
			<Value>Q == 6000 ; S_1020</Value>
			<Value>Q == 8000 ; Q1500</Value>
			<Value>Q == 10000 ; S_2000</Value>
			<Value>Q == 12000 ; S_2000</Value>
			<Value>Q == 13000 ; Q3000</Value>
			<Value>Q == 14000 ; Q3000</Value>
			<Value>Q == 15000 ; Q3000</Value>
			<Value>Q == 16000 ; Q3000</Value>
			<Value>Q == 17000 ; S_4000</Value>
```

```
                    <Value>Q == 18000 ; S_4000</Value>
                    <Value>Q == 20000 ; S_4000</Value>
                    <Value>Q == 24000 ; Q5000</Value>
             </Values>
      </Variable>
</Variables>
```

Hiermee is de definitie van het PDB gereed.

De volgende stap is het bepalen van de benodigde templates. Het primaire invoerbestand voor D-Hydro is een MDU-bestand. We onderscheiden in deze user example één invoerbestand voor D-Hydro.

Template van het type 'inputfile'

```
<Templates>
      <Template>
            <Type>inputfile</Type>
            <File>RIJN.mdu</File>
            <Update>true</Update>
      </Template>
      …
</Templates>
```

Dit templatebestand bevat SGWM Keys die bijgewerkt moeten worden. Daarom zetten we 'Update' op 'true'.
In D-Hydro wordt gebruik gemaakt van een DIMR-configuratie. We onderscheiden in deze user example slechts één DIMR-configuratie (parallel rekeningen met RTC).

Templates van het type 'general' voor de DIMR-configuratie

```
<Templates>
      <Template>
            <Type>general</Type>
            <File>dimr_config</File>
            <Update>true</Update>
      </Template>
      …
</Templates>
```

Het DIMR-templatebestand is geen inputfile van D-Hydro en valt daarom onder het type 'general'. Er zijn SGWM Keys opgenomen in dit templatebestand, dus is 'Edit' gelijk aan 'true'.

De D-Hydro som maakt verder gebruik van een netCDF bestand. Deze template zijn van het type 'general' en bevat geen SGWM Keys.

Overige templates van het type 'general'

```
<Templates>
      …
      <Template>
            <Type>general</Type>
            <File>rijn-hr2023_6-v1a_net.nc</File>
            <Update>false</Update>
      </Template>
      …
</Templates>
```

Uitgaande van het feit dat de berekeningen gedaan worden op het Modellen-Platform van SSC-Campus, maken we in elke som gebruik van een bestand 'options.sh'. In dit bestand is vastgelegd hoe de netwerk schijven/mappen

'gemount' moeten worden aan de docker met de D-Hydro software. Dit templatebestand bevat geen SGWM Keys en wordt niet als invoerbestand voor D-Hydro gebruikt, noch als lsf of bsub bestand. In de definitie in het PDB wordt dit bestand dan als 'general' getypeerd en 'Update' is 'false'.

Template van het type 'general'

```
<Templates>
	…
	<Template>
		<Type>general</Type>
		<File>options.sh</File>
		<Update>false</Update>
	</Template>
</Templates>
```

Voor het uitvoeren van een D-Hydro berekening is een lsf jobscript gedefinieerd. Dit bestand ziet er als volgt uit:

LSF Jobscript template D-Hydro

```
#!/bin/bash
#$ -cwd

#BSUB -J rijn_Q{TypeAfvoer}{Q:5}
#BSUB -oo log rijn_Q{TypeAfvoer}{Q:5}.o
#BSUB -eo log rijn_Q{TypeAfvoer}{Q:5}.e
#BSUB -n {N}
#BSUB -R "span[hosts=1]"

cd rijn_Q{TypeAfvoer}{Q:5}

echo "Run parallel preprocessing"
export
LD_LIBRARY_PATH=/opt/delft3dfm_latest/lnx64/lib:$LD_LIBRARY_PATH
/opt/delft3dfm_latest/lnx64/bin/dflowfm --
partition:ndomains=$(($(echo "'{PARDocker}'" | grep -o " " | wc -
l)+1)):icgsolver=6 rijn_Q{TypeAfvoer}{Q:5}.mdu
# Wait for all jobs to complete
wait


ulimit -s unlimited

echo "Run parallel"
/opt/delft3dfm_latest/lnx64/bin/run_dimr.sh --debug 9 --
dockerparallel -c $(($(echo "'{PARDocker}'" | grep -o " " | wc -
l)+1)) --D3D_HOME /opt/delft3dfm_latest/lnx64 -m dimr_config.xml

echo "Run parallel postprocessing"
cd results

ls *_map.nc > list_map.txt
/opt/delft3dfm_latest/lnx64/bin/run_dfmoutput.sh -- mapmerge --
listfile list_map.txt --outfile rijn_Q{TypeAfvoer}{Q:5}_map.nc
ls *_fou.nc > list_fou.txt
/opt/delft3dfm_latest/lnx64/bin/run_dfmoutput.sh -- mapmerge --
listfile list_fou.txt --outfile rijn_Q{TypeAfvoer}{Q:5}_fou.nc
ls *_clm_*.nc > list_clm.txt
/opt/delft3dfm_latest/lnx64/bin/run_dfmoutput.sh -- mapmerge --
listfile list_clm.txt --outfile rijn_Q{TypeAfvoer}{Q:5}_clm.nc

/opt/delft3dfm_latest/lnx64/bin/run_dfmoutput.sh -- max25 --
infile rijn_Q{TypeAfvoer}{Q:5}_0000_his.nc --outfile
rijn_Q{TypeAfvoer}{Q:5}_max13_v1.txt
/opt/delft3dfm_latest/lnx64/bin/run_dfmoutput.sh --
max_running_mean --infile rijn_Q{TypeAfvoer}{Q:5}_0000_his.nc --
outfile rijn_Q{TypeAfvoer}{Q:5}_max13_v2.txt
```

```
/opt/delft3dfm_latest/lnx64/bin/run_dfmoutput.sh --
max_running_mean --infile rijn_Q{TypeAfvoer}{Q:5}_0000_his.nc --
outfile rijn_Q{TypeAfvoer}{Q:5}_max13_v2_ws13.txt --filterlength
13
/opt/delft3dfm_latest/lnx64/bin/run_dfmoutput.sh --
max_running_mean --infile rijn_Q{TypeAfvoer}{Q:5}_0000_his.nc --
outfile rijn_Q{TypeAfvoer}{Q:5}_max13_v2_ws25.txt --filterlength
25

cat rijn_Q{TypeAfvoer}{Q:5}_0000_numlimdt.xyz
rijn_Q{TypeAfvoer}{Q:5}_0001_numlimdt.xyz
rijn_Q{TypeAfvoer}{Q:5}_0002_numlimdt.xyz
rijn_Q{TypeAfvoer}{Q:5}_0003_numlimdt.xyz
rijn_Q{TypeAfvoer}{Q:5}_0004_numlimdt.xyz
rijn_Q{TypeAfvoer}{Q:5}_0005_numlimdt.xyz
rijn_Q{TypeAfvoer}{Q:5}_0006_numlimdt.xyz
rijn_Q{TypeAfvoer}{Q:5}_0007_numlimdt.xyz >
rijn_Q{TypeAfvoer}{Q:5}_numlimdt.xyz

cd ..
```

De eerste regel bevat een shell commando, gevolgd door 3 regels waarin parameters van het Jobscript worden gedefinieerd. Hierin wordt gebruik gemaakt van SGWM Keys (tussen accoladen). Daarna volg het script voor het uitvoeren van de berekening en de verwijzing naar het DIMR configuratie-bestand.

Dit Jobscript is een templatebestand dat we als volgt definiëren in het MDB:

Template van het type 'lsf' (voor het D-Hydro jobscript)

```
<Templates>
        …
        <Template>
                <Type>lsf</Type>
                <File>lsf.job.script.MPSSC</File>
                <Update>true</Update>
        </Template>
</Templates>
```

'Update' is gelijk aan 'true' omdat in het template gebruik is gemaakt van SGWM Keys.

Voor het toevoegen van sommen aan de wachtrij van LSF op het Modellen-Platform van het NWM, wordt gebruik gemaakt van bsub commando's. Deze zijn eveneens opgenomen in een templatebestand. Dit bestand ziet er als volgt uit:

LSF bsub template D-Hydro

```
cd rijn_Q{TypeAfvoer}{Q:5}
bsub -env "LSB_CONTAINER_IMAGE={ApplicationContainer}" -app
docker -q {Queue} < lsf.job.script.MPSSC cd ..
cd ..
```

Bovengenoemde template wordt door SGWM voor elke som gekopieerd een toegevoegd aan één totaal bsub bestand. Voor elke som wordt eerst (vanuit de map waarin het bsub bestand wordt gestart; de map '…/computations') naar de map met de betreffende somID genavigeerd. In het templatebestand wordt hiervoor de somID als SGWM Key gebruikt (tussen accoladen). Daarna wordt het bsub commando van de som gedefinieerd, waarin de naam van de

Docker container met D-Hydro wordt meegegeven (ook een SGWM Key), een queue parameter voor de prioriteit van de som en de verwijzing naar het Jobscript bestand. Nadat een som via het bsub commando is toegevoegd aan de wachtrij, wordt uit de map van de som genavigeerd, terug naar de map '…/computations'.

Dit bsub-commando template definiëren we als volgt in het MDB:

Template van het type 'bsub'

```
<Templates>
        …
        <Template>
                <Type>bsub</Type>
                <File shell="#!/bin/bash">bsub_file_MPSSC.sh</File>
                <Update>true</Update>
        </Template>
</Templates>
```

Merk op dat in de verwijzing naar het templatebestand gebruik is gemaakt van een de 'shell' optie. Hiermee zorgen we ervoor dat bij het samenstellen van het bsub bestand voor alle sommen, de eerste regel wordt gevuld met het shell commando zoals tussen de dubbel quotes is opgenomen. Als we dit zouden toevoegen aan het bsub templatebestand zou dit shell commando voor elke som herhaald worden, wat niet de bedoeling is. Ook hier is 'Update' gelijk aan 'true', aangezien er gebruik is gemaakt van SGWM Keys in het templatebestand, zoals 'Q', 'Queue' en 'ApplicationContainer'.

Nadat alle templatebestanden gedefinieerd zijn in het MDB, moet gedefinieerd worden of er een subfolder gebruikt wordt voor de invoerbestanden en wat de naam is van het D-Hydro input bestand. In dit voorbeeld hebben we gekozen om geen subfolder aan te laten maken. De naam van het D-Hydro invoerbestand is gelijk aan de somID. Dit definiëren we als volgt in het MDB:

Definitie van model input bestand

```
<Input>
        <Folder>input</Folder>
        <File>rijn_Q{AfvoerType}{Q:5}.mdu</File>
</Input>
```

Als laatste onderdeel van de modelschematisatie in het MDB definiëren we de zogenoemde 'states'. We onderscheiden in dit voorbeeld 3 states van elke som: 'invoer gereed', 'som gereed' en 'fout opgetreden'. Voor elke state definiëren we de naam van de state, de naam van het bestand waarin naar het voorkomen van een specifieke tekst moet worden gezocht en folder waarin het bestand wordt verwacht.

De definitie van de states in het PDB ziet er als volgt uit:

Definitie van states

```
<States>
	<State>
		<Name>invoer gereed</Name>
		<Text></Text>
		<File>*.mdu</File>
		<Folder> </Folder>
		<Progress>false</Progress>
	</State>
	<State>
		<Name>som gereed</Name>
		<Text>Successfully completed.</Text>
		<File>*.o.*</File>
		<Folder></Folder>
		<Progress>false</Progress>
	</State>
	<State>
		<Name>fout opgetreden</Name>
		<Text>exit code</Text>
		<File>*.o*</File>
		<Folder> </Folder>
		<Progress>false</Progress>
	</State>
</States>
```

De volgorde van de states bepaalt hoe SGWM de status van een som bepaald. In dit geval zal SGWM eerst controleren of er een bestand met de extensie 'mdu' aanwezig is in de map van elke som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand gezocht. Als het bestand met de extensie 'mdu' voorkomt krijgt de som de status 'invoer gereed'. Vervolgens controleert SGWM of in de map van elke som een bestand met '.o' in de naam voorkomt en of in dit bestand de tekst 'Successfully completed.' voorkomt. Als dat zo dan krijgt de som (alsnog) de status 'som gereed'. Tot slot controleert SGWM of in de map van elke som een bestand met '.e' in de naam voorkomt en of dit bestand de tekst 'Error' bevat. Als dat het geval is dan krijgt de som (alsnog) de status 'fout opgetreden'. Merk op dat het gebruik van de bestanden '.o' en '.e' een relatie heeft met de definitie van de output en error bestanden in de parameters in het D-Hydro Jobscript.

Tot slot zijn in het MDB vier rekeninstellingen gedefinieerd. Allereerst een rekeninstelling voor een queue parameter in LSF. Deze kan meerdere gedefinieerde tekstwaarden aannemen, dus definiëren we deze als combo box. Omdat we de queue parameter willen kunnen aanpassen in de gebruikersschil van SGWM zetten we 'Edit' op 'true'. De tweede rekeninstelling bevat de informatie voor de D-Hydro Docker Container op het ModellenPlatform bij NWM. Dit kunnen verschillende versies zijn en daarom ook als combo box gedefinieerd en dus staat 'Edit' op 'true'. De derde en vierde rekeninstelling zijn er om het aantal parallelle sommen en cores te definiëren.

Dit ziet er dan als volgt uit in het MDB:

Definitie van rekeninstellingen in het PDB

```
<CalculationSettings>
       <CalculationSetting>
            <Name>Queue</Name>
            <Key>Queue</Key>
            <Type>combo</Type>
            <Items>RWS_meren_dhydro;RWS_meren_swan;
                RWS_maas_dhydro;RWS_normal;RWS_medium;RWS_large;
                RWS_rijn_dhydro</Items>
            <Value>RWS_rijn_dhydro</Value>
            <Edit>true</Edit>
        </CalculationSetting>
        <CalculationSetting>
            <Name>Application Container</Name>
            <Key>ApplicationContainer</Key>
            <Type>combo</Type>
            <Items>deltares/delft3dfm:2021.04_1.0.0;
                deltares/delft3dfm:2022.01patch01_1.0.0</Items>
           <Value>deltares/delft3dfm:2022.01patch01_1.0.0</Value>
            <Edit>true</Edit>
        </CalculationSetting>
        <CalculationSetting>
            <Name>Docker parallel</Name>
            <Key>PARDocker</Key>
            <Type>combo</Type>
            <Items>0;0 1;0 1 2;0 1 2 3;0 1 2 3 4 5 6 7;</Items>
            <Value>0 1 2 3 4 5 6 7</Value>
            <Edit>true</Edit>
        </CalculationSetting>
        <CalculationSetting>
            <Name>Parallel</Name>
            <Key>N</Key>
            <Type>combo</Type>
            <Items>0;2;4;8;</Items>
            <Value>8</Value>
            <Edit>true</Edit>
        </CalculationSetting>
</CalculationSettings>
```

Hiermee is de definitie van het MDB volledig.

De map met templatebestanden ('.../computations/templates') bevat 6 templates, waarvan het jobscript, het bsub commando script, options.sh en de 1 DIMR config-bestand al besproken zijn. Het MDU-bestand is het invoerbestand van D-Hydro en bevat 1 SGWM Key, een verwijzing naar een bestand met randvoorwaarden. Omdat het bestand verder zeer uitgebreid is wordt dit bestand hier niet weergegeven. Het laatste bestand (een NetCDF bestand) wordt alleen naar elke som map gekopieerd.

2.5.4 Rekenmodel Sobek

In dit voorbeeld wordt gebruik gemaakt van SOBEK versie 3.7.21.50810 die lokaal beschikbaar is gemaakt. Daarnaast beschouwen we alleen de RijnMaasmonding.

De basis van het PDB wordt:

Basis Sobek PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
	<Name>SOBEK-RMM</Name>
	<Instrumentarium>nwm-ve</Instrumentarium>
	<ModelSoftwarePackage>
		<Name>SOBEK</Name>
		<Version>3.7.21</Version>
	</ModelSoftwarePackage>
	<Modelschematisation></Modelschematisation>
</PDB>
```

Er moeten sommen gemaakt worden voor 9 afvoeren op de Rijn en Maas en voor 6 zeewaterstanden (de zogenoemde 54-sommen set). We streven naar een somID die er als volgt uit ziet: 'Q1_H1_U1_D1':

- Q staat voor de afvoer op Rijn (en Maas) in m^3/s.
- H staat voor de zeewaterstand in m+NAP. Daaraan gekoppeld zijn de waarde van U (windsnelheid) en de D (windrichting).

We beschouwen verder voor elke combinatie van zeewaterstand en windsnelheid, één windrichting en dan met id gelijk aan 1. Deze wordt niet als parameter, noch als variabele gedefinieerd, maar is wel gebruikt in de somID. De somID wordt dan als volgt gedefinieerd in het PDB:

SomID in PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
	<Name>SOBEK-RMM</Name>
	<Instrumentarium>nwm-ve</Instrumentarium>
	<ModelSoftwarePackage></ModelSoftwarePackage>
	<Modelschematisation>
		<Name>sobek-rmm-vozo-j15_5-v3</Name>
		<Folder>…</Folder>
		<Geometry>
			<Folder>geometry</Folder>
		</Geometry>
		<BoundaryCondition>
			<Folder>boundary_conditions</Folder>
		</BoundaryCondition>
		<Computation>
			<Folder>computations</Folder>
			<Id>Q{Q}_H{H}_U{U}_D1</id>
			…
		</Computation>
	</Modelschematisation>
</PDB>
```

De afvoer en zeewaterstand worden parameters in het PDB. Voor deze parameters wordt niet de echte waarde gebruikt maar een id (overeenkomstig een waarde uit de 54 sommenset).

Dit resulteert in de volgende definitie in het PDB:

Definitie parameters SOBEK in PDB

```
<Parameters>
	<Parameter>
		<Name>Afvoer Rijn en Maas (id)</Name>
		<Key>Q</Key>
		<Type>float</Type>
		<Values>
			<Value>1</Value>
			<Value>2</Value>
			<Value>3</Value>
			<Value>4</Value>
			<Value>5</Value>
			<Value>6</Value>
			<Value>7</Value>
			<Value>8</Value>
			<Value>9</Value>
		</Values>
	</Parameter>
	<Parameter>
		<Name>Zeewaterstand (id)</Name>
		<Key>H</Key>
		<Type>string</Type>
		<Values>
			<Value>1</Value>
			<Value>2</Value>
			<Value>3</Value>
			<Value>4</Value>
			<Value>5</Value>
			<Value>6</Value>
		</Values>
	</Parameter>
</Parameters>
```

De windsnelheid is direct gekoppeld aan de zeewaterstand. Voor elke zeewaterstand is ook een winsnelheid gedefinieerd. Deze koppeling maken we aan via de definitie van variabelen in het PDB. Ook hier is niet de echte waarde van de winsnelheid gebruikt maar een id.

Definitie variabelen SOBEK in PDB

```
<Variables>
	<Variable>
		<Name>Windsnelheid id</Name>
		<Key>U</Key>
		<Type>float</Type>
		<Values>
			<Value>H == 1 ; 1</Value>
			<Value>H == 2 ; 2</Value>
			<Value>H == 3 ; 3</Value>
			<Value>H == 4 ; 4</Value>
			<Value>H == 5 ; 5</Value>
			<Value>H == 6 ; 6</Value>
		</Value>
	</Variables>
</Variables>
```

Hiermee is de definitie van het PDB gereed.

De volgende stap is het bepalen van de benodigde templates. De wijze waarop wij deze 54 sommenset aan Sobek aanbieden, vereist niet dat we een invoerbestand nodig hebben (zie ook de beschrijving van het jobscript verderop in deze paragraaf). Het MDB vereist echter wel altijd 1 template van

het type 'inputfile'. Daarom hebben we ervoor gekozen om een dummy bestand als invoerbestand te definiëren in het MDB. Dit bestand is helemaal leeg en bevat dus ook geen SGWM Keys. Daarom hebben we 'Edit' of 'false' gezet.

Template van het type 'inputfile'

```
<Templates>
	<Template>
		<Type>inputfile</Type>
		<File>dummy.txt</File>
		<Update>false</Update>
	</Template>
	…
</Templates>
```

Voor het uitvoeren van de Sobek berekening is een lsf jobscript gedefinieerd. Dit bestand ziet er als volgt uit:

LSF Jobscript template Sobek

```
rem set OMP_NUM_THREADS=2
xcopy /Y
..\..\..\bc_files\Q{Q}_H{H}_U{U}_D1\BoundaryConditions.bc
cd ..\..\..\model\dflow1d
xcopy /Y ..\..\bc_files\Q{Q}_H{H}_U{U}_D1\BoundaryConditions.bc
rem delete log prior to new simulation; this will guarantee that
the log represents the last run
del sobek.log
cd ..
call "c:\Program Files (x86)\Deltares\SOBEK
(3.7.21.50810)\plugins\DeltaShell.Dimr\kernels\x64\dimr\scripts\r
un_dimr.bat" sobek-rmm-vozo-j15_5-v3.xml
cd ..\computations\Q{Q}_H{H}_U{U}_D1
move /Y ..\..\..\model\*.nc *.nc
move /Y ..\..\..\model\dflow1d\sobek.log
rem output is copied. Move will (re)move the output directory;
without this directory SOBEK3 wil NOT write the output files
xcopy /Y/S ..\..\..\model\dflow1d\output output\
```

In dit Jobscript worden (voor elke som) achtereenvolgens de volgende stappen gedaan:

- Het kopiëren van de bestanden met randvoorwaarden(tijdreeksen) van de betreffende som uit de map met randvoorwaarden 'bc-files' naar de som directory.
- Het kopiëren van de bestanden met randvoorwaarden(tijdreeksen) van de betreffende som uit de map met randvoorwaarden 'bc-files' naar de model-folder van Sobek (in de map 'model/dflow1d').
- Het verwijderen van een eventuele bestaande Sobek log.
- Het aanroepen van de Sobek software.
- Het kopiëren/verplaatsen van de resultatenbestanden uit de model-folder van Sobek naar de map van de betreffende som.

Dit Jobscript is een templatebestand dat we als volgt definiëren in het MDB:

Template van het type 'lsf' (voor het Sobek jobscript)

```
<Templates>
	…
	<Template>
		<Type>lsf</Type>
		<File>execute_single_run.bat</File>
		<Update>true</Update>
	</Template>
</Templates>
```

'Update' is gelijk aan 'true' omdat in het template van het jobscript gebruik is gemaakt van SGWM Keys. Merk verder op dat hier gebruik is gemaakt van een bat bestand omdat we Sobek op een lokale computer gaan gebruiken. Om meerdere sommen achter elkaar te kunnen uitvoeren, maken we een bsub commando script aan. In dit bestand wordt per som een call gedaan naar het jobscript (bat bestand). Dit bestand ziet er als volgt uit:

LSF bsub template Sobek

```
cd Q{Q}_H{H}_U{U}_D1/input
call execute_single_run.bat
cd ..
cd ..
```

Bovengenoemde template wordt door SGWM voor elke som gekopieerd een toegevoegd aan één totaal bsub bestand.

Dit wordt ook een templatebestand en definiëren we als volgt in het MDB:

Template van het type 'bsub'

```
<Templates>
	…
	<Template>
		<Type>bsub</Type>
		<File> shell="set OMP_NUM_THREADS=2">
			bsub_joblist.bat</File>
		<Update>true</Update>
	</Template>
</Templates>
```

Hier is 'Update' gelijk aan 'true', aangezien er gebruik is gemaakt van SGWM Keys in het templatebestand, zoals 'ID'.

Nadat alle templatebestanden gedefinieerd zijn in het MDB, moet gedefinieerd worden of er een subfolder gebruikt wordt voor de invoerbestanden en wat de naam is van het D-Hydro input bestand. In dit voorbeeld hebben we gekozen om de subfolder 'input' aan te laten maken. We hanteren de naam van het dummy bestand als fictief invoerbestand voor Sobek. Dit bestand wordt in de berekeningen niet gebruikt. Dit definiëren we als volgt in het MDB:

Definitie van model input bestand

```
<Input>
	<Folder>input</Folder>
	<File>dummy.txt</File>
</Input>
```

Als laatste onderdeel van de modelschematisatie in het MDB definiëren we de zogenoemde 'states'. We onderscheiden in dit voorbeeld 3 states van elke som: 'invoer klaar', 'som gereed' en 'fout opgetreden'. Voor elke state definiëren we de naam van de state, de naam van het bestand waarin naar het voorkomen van een specifieke tekst moet worden gezocht en folder waarin het bestand wordt verwacht. De definitie van de states in het MDB ziet er als volgt uit:

Definitie van states

```
<States>
	<State>
		<Name>invoer klaar</Name>
		<Text></Text>
		<File>dummy.txt</File>
		<Folder>input</Folder>
		<Progress>false</Progress>
	</State>
	<State>
		<Name>som gereed</Name>
		<Text>** Normal End of Sobeksim</Text>
		<File>sobek.log</File>
		<Folder>output</Folder>
		<Progress>true</Progress>
	</State>
	<State>
		<Name>fout opgetreden</Name>
		<Text>** FATAL</Text>
		<File>sobek.log</File>
		<Folder>output</Folder>
		<Progress>false</Progress>
	</State>
</States>
```

De volgorde van de states bepaalt hoe SGWM de status van een som bepaald. In dit geval zal SGWM eerst controleren of er een bestand 'dummy.txt' aanwezig is in de subfolder 'input' van elke som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand gezocht. Als het bestand voorkomt krijgt de som de status 'invoer gereed'. Vervolgens controleert SGWM of subfolder 'output' in de rootmap van elke som een bestand 'sobek.log' voorkomt en of dit bestand de tekst '** Normal End of Sobeksim.' bevat. Als dat zo is dan krijgt de som (alsnog) de status 'som gereed'. Tot slot controleert SGWM hetzelfde bestand de tekst 'Error' voorkomt. Als dat het geval is dan krijgt de som (alsnog) de status 'fout opgetreden'.

Tot slot zijn in het MDB een dummy rekeninstelling gedefinieerd. Er zijn namelijk geen rekeninstelling nodig, maar het PDB vereist minimaal rekeninstelling. Dit ziet er dan als volgt uit in het PDB:

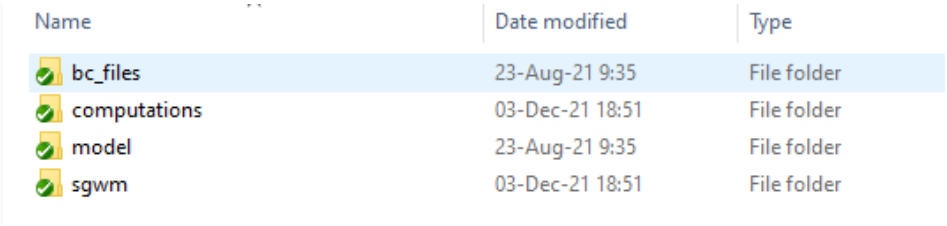
Definitie van rekeninstellingen in het PDB

```
<CalculationSettings>
	<CalculationSetting>
		<Name>Dummy</Name>
		<Key>Dummy</Key>
		<Type>string</Type>
		<Value>Dummy</Value>
		<Edit>false</Edit>
	</CalculationSetting>
</CalculationSettings>
```

Omdat deze configuratie (voor Sobek) geen gebruik maakt van de standaard mappen-indeling van SGWM ('boundary_conditions', 'computations', 'geometry' en 'initial_conditions') laten we het deel van het MDB zien waarin de padnamen zijn aangepast, conform de opzet in de modelschematisatie zelf. De tag <InitialConditons> is helemaal weggelaten en voor 'geometry' is 'model' gebruikt en voor 'boundary_conditions', 'bc_files'.

Afwijkende padnamen

```
<Geometry>
        <Folder>model</Folder>
</Geometry>
<BoundaryCondition>
        <Folder>bc_files</Folder>
</BoundaryCondition>
```



Hiermee is de definitie van het MDB volledig.

De map met template \bestanden ('.../computations/templates') bevat slechts 3 templates, waarvan het jobscript en het bsub commando script al besproken zijn. De overgebleven template betreft het dummy invoerbestand voor Sobek, dat verder leeg is.

2.5.5 Rekenmodel Hydra-NL

In dit voorbeeld wordt gebruik gemaakt van het rekenhart van Hydra_Zoet versie 2.4.1 die lokaal beschikbaar is gemaakt. Daarnaast beschouwen we alleen de RijnMaasmonding. De basis van het PDB wordt:

Basis Hydra-NL PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
        <Name>Hydra-NL-RMM</Name>
        <Instrumentarium>nwm-ve</Instrumentarium>
        <ModelSoftwarePackage>
                <Name>Hydra-NL-Zoet (rekenhart)</Name>
                <Version>2.4.1 (mei 2018)</Version>
        </ModelSoftwarePackage>
        <Modelschematisation></Modelschematisation>
</PDB>
```

Er moeten sommen gemaakt worden voor 5 locaties van 2 type berekeningen. We streven naar een somID die er als volgt uit ziet: 'Locatie DK_1_22-1_dk_00291TypeWS':

- Locatie staat voor de locatienaam in Hydra-NL.
- Type staat voor het berekeningstype in Hydra-NL, bijvoorbeeld 'WS' voor een waterstandsberekening of 'HBN' voor een HBN-berekening.

De somID wordt dan als volgt gedefinieerd in het PDB:

SomID in PDB

```
<?xml version="1.0" encoding="UTF-8"?>
<PDB>
	<Name>Hydra-NL-RMM</Name>
	<Instrumentarium>nwm-ve</Instrumentarium>
	<ModelSoftwarePackage></ModelSoftwarePackage>
	<Modelschematisation>
		<Name>hydra-wbi2017-v1</Name>
		<Folder>…</Folder>
		<Geometry>
			<Folder>software</Folder>
		</Geometry>
		<BoundaryCondition>
			<Folder>data</Folder>
		</BoundaryCondition>
		<InitialCondition>
			<Folder>databases</Folder>
		</InitialCondition>
		<Computation>
			<Folder>computations</Folder>
			<Id>Locatie{Locatie}Type{Type}</id>
			…
		</Computation>
	</Modelschematisation>
</PDB>
```

Voor elk wordt een parameter aangemaakt in het PDB.

Parameters Hydra-NL in PDB

```
<Parameters>
	<Parameter>
		<Name>Locatienaam in Hydra-NL</Name>
		<Code>Locatie</Code>
		<Type>string</Type>
		<Values>
			<Value>DK_1_22-1_dk_00291</Value>
			<Value>HD_1_34-2_dk_00345</Value>
		</Values>
	</Parameter>
	<Parameter>
		<Name>Type Hydra-NL berekening</Name>
		<Code>Type</Code>
		<Type>string</Type>
		<Values>
			<Value>WS</Value>
			<Value>HBN_1</Value>
			<Value>HBN_10</Value>
			<Value>Debiet</Value>
			<Value>Steenzet_2</Value>
			<Value>Steenzet_3</Value>
			<Value>Breuksteen_2</Value>
			<Value>Breuksteen_3</Value>
			<Value>Grasoploop_2.5</Value>
			<Value>Hm0</Value>
			<Value>Tspec</Value>
			<Value>Tp</Value>
		</Values>
	</Parameter>
</Parameters>
```

Elke locatie moet zowel een x- als een y-coördinaat hebben. Hiertoe hebben we in het PDB twee variabelen, respectievelijk ‘Xcoordinaat’ en ‘Ycoordinaat’ gedefinieerd.

Variabele X- en Ycoordinaat Hydra-NL in PDB

```
<Variables>
	<Variable>
		<Name>x-coordinaat</Name>
		<Key>Xcoordinaat</Key>
		<Type>float</Type>
		<Values>
			<Value>Locatie == 'DK_1_22-1_dk_00291' ;
				102629</Value>
			<Value>Locatie == 'HD_1_34-2_dk_00345' ;
				89297</Value>
		</Values>
	</Variable>
	<Variable>
		<Name>y-coordinaat</Name>
		<Key>Ycoordinaat</Key>
		…
	</Variable>
	…
</Variables>
```

Daarnaast krijgt elke locatie een Hydra-NL profiel toegekend. Ook hiervoor hebben we variabele 'Profielnaam' in het PDB voor gedefinieerd.

Variabele Profielnaam Hydra-NL in PDB

```
<Variables>
	…
	<Variable>
		<Name>Profielnaam</Name>
		<Key>Profielnaam</Key>
		<Type>string</Type>
		<Values>
			<Value>Locatie == 'DK_1_22-1_dk_00291' ;
				Bermprofiel</Value>
			<Value>Locatie == 'HD_1_34-2_dk_00345' ;
				Profiel met flauw talud</Value>
		</Values>
	</Variable>
	…
</Variables>
```

Vervolgens zijn er per rekenmethode een fors aantal aanvullende instellingen benodigd. Deze zijn opgenomen in onderstaande tabel.

Tabel 6 Invoer per type berekening in Hydra-NL

Type berekening	Volg nr	Type nr	QRC	NI-VEAU	PAR-TYPE	BEKLE-DING	A	B	C
WS	1	0	1	0	0	geen	0	0	0
HBN_1	2	1	0.001	0	0	geen	0	0	0
HBN_10	3	1	0.01	0	0	geen	0	0	0
Debiet	4	2	1	0	0	geen	0	0	0
Steenzet_2	5	4	1	2	0	Steenzetting betonzuilen (normale golfsteilheid)	1	0.4	0.8
Steenzet_3	6	4	1	3	0	Steenzetting betonzuilen (normale golfsteilheid)	1	0.4	0.8
Breuksteen_2	7	4	1	2	5	Breuksteen (normale golfsteilheid)	1	0.67	1.1
Breuksteen_3	8	4	1	3	5	Breuksteen (normale golfsteilheid)	1	0.67	1.1

Type berekening	Volg nr	Type nr	QRC	NI-VEAU	PAR-TYPE	BEKLE-DING	A	B	C
Grasoploop_2.5	9	4	1	2.5	9	Grasmat oploopzone	1	1.7	0.3
Hm0	10	6	1	0	0	geen	0	0	0
Tspec	11	7	1	0	0	geen	0	0	0
Tp	12	8	1	0	0	geen	0	0	0

Voor elk van de kolommen uit bovenstaande tabel is een variabele gedefinieerd in het PDB.

Overige variabelen in PDB

```
<Variables>
        …
        <Variable>
                <Name>Volgnummer type berekening</Name>
                <Key>VolgnummerBerekeningType</Key>
                <Type>float</Type>
                <Values>
                        <Value>Type == 'WS' ; 1</Value>
                        <Value>Type == 'HBN_1' ; 2</Value>
                        <Value>Type == 'HBN_10' ; 3</Value>
                        <Value>Type == 'Debiet' ; 4</Value>
                        <Value>Type == 'Steenzet_2' ; 5</Value>
                        <Value>Type == 'Steenzet_3' ; 6</Value>
                        <Value>Type == 'Breuksteen_2' ; 7</Value>
                        <Value>Type == 'Breuksteen_3' ; 8</Value>
                        <Value>Type == 'Grasoploop_2.5' ; 9</Value>
                        <Value>Type == 'Hm0' ; 10</Value>
                        <Value>Type == 'Tspec' ; 11</Value>
                        <Value>Type == 'Tp' ; 12</Value>
                </Values>
        </Variable>
        …
</Variables>
```

Hiermee zijn alle variabelen en parameters voor deze user example gedefinieerd.

De volgende stap is het bepalen van de benodigde templates. Het primaire invoerbestand voor Hydra-NL is een hyd-bestand. Voor elke som is er één hyd-invoerbestand benodigd, dit wordt dus een template van het type 'inputfile'. Dit templatebestand bevat SGWM Keys die bijgewerkt moeten worden. Daarom zetten we 'Update' op 'true'.

Template van het type 'inputfile'

```
<Templates>
        <Template>
                <Type>inputfile</Type>
                <File>Hydra-invoer.hyd</File>
                <Update>true</Update>
        </Template>
        …
</Templates>
```

Voor het uitvoeren van een Hydra berekening is een lsf jobscript gedefinieerd. Dit bestand ziet er als volgt uit:

LSF Jobscript template Hydra-NL

```
mkdir ..\uitvoer
..\..\..\software\Rekenhart_Zoet.exe invoer.hyd
```

De opzet is zeer eenvoudig. Er wordt een aparte folder voor de resultaatbestanden aangemaakt en het rekenhart van Hydra-Zoet wordt aangeroepen waarbij het invoerbestand wordt meegegeven.

Dit Jobscript is een templatebestand dat we als volgt definiëren in het PDB:

Template van het type 'lsf' (voor het hydra-NL jobscript)

```
<Templates>
	…
	<Template>
		<Type>lsf</Type>
		<File>lsf.job.HYDRA.HKV.bat</File>
		<Update>true</Update>
	</Template>
</Templates>
```

'Update' is gelijk aan 'False' omdat in het template geen gebruik is gemaakt van SGWM Keys. Merk verder op dat hier gebruik is gemaakt van een bat bestand omdat we Hydra-NL op een lokale computer gaan gebruiken. Om meerdere sommen achter elkaar te kunnen uitvoeren, maken we een bsub commando script aan. In dit bestand wordt per som een call gedaan naar het jobscript (bat bestand).

Dit bestand ziet er als volgt uit:

LSF bsub template Hydra-NL

```
cd {Locatie{Locatie}Type{Type}/invoer
call lsf.job.script.HYDRA.HKV.bat
cd ..
cd ..
```

Bovengenoemde template wordt door SGWM voor elke som gekopieerd een toegevoegd aan één totaal bsub bestand. Dit wordt ook een templatebestand en definiëren we als volgt in het MDB:

Template van het type 'bsub'

```
<Templates>
	…
	<Template>
		<Type>bsub</Type>
		<File>bsub_file_HYDRA.HKV.bat</File>
		<Update>true</Update>
	</Template>
</Templates>
```

Hier is 'Update' gelijk aan 'true', aangezien er gebruik is gemaakt van SGWM Keys in het templatebestand.

Nadat alle templatebestanden gedefinieerd zijn in het MDB, moet gedefinieerd worden of er een subfolder gebruikt wordt voor de invoerbestanden en wat de naam is van het D-Hydro input bestand. In dit voorbeeld hebben we gekozen om de subfolder 'invoer' aan te laten maken. We passen de naam van het invoerbestand iets aan door 'Hydra_' te verwijderen. Dit bestand wordt in de berekeningen niet gebruikt.

Dit definiëren we als volgt in het MDB:

Definitie van model input bestand

```
<Input>
	<Folder>invoer</Folder>
	<File>invoer.hyd</File>
</Input>
```

Als laatste onderdeel van de modelschematisatie in het MDB definiëren we de zogenoemde 'states'. We onderscheiden in dit voorbeeld 3 states van elke som: 'invoer gereed', 'som gestart' en 'som gereed'. Voor elke state definiëren we de naam van de state, de naam van het bestand waarin naar het voorkomen van een specifieke tekst moet worden gezocht en folder waarin het bestand wordt verwacht. De definitie van de states in het MDB ziet er als volgt uit:

Definitie van states

```
<States>
	<State>
		<Name>invoer gereed</Name>
		<Text></Text>
		<File>invoer.hyd</File>
		<Folder>invoer</Folder>
		<Progress>false</Progress>
	</State>
	<State>
		<Name>som gestart</Name>
		<Text></Text>
		<File>uitvoer.html</File>
		<Folder>uitvoer</Folder>
		<Progress>false</Progress>
	</State>
	<State>
		<Name>som gereed</Name>
		<Text>Hydra-NL</Text>
		<File>uitvoer.html</File>
		<Folder>uitvoer</Folder>
		<Progress>true</Progress>
</States>
```

De volgorde van de states bepaalt hoe SGWM de status van een som bepaald. In dit geval zal SGWM eerst controleren of er een bestand 'invoer.hyd' aanwezig is in de subfolder 'invoer' van elke som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand gezocht. Als het bestand voorkomt krijgt de som de status 'invoer gereed'. Vervolgens wordt gecontroleerd of er een bestand 'uitvoer.html' voorkomt in de subfolder 'uitvoer' in de rootmap van elke som. Omdat we geen tekst hebben aangeduid, wordt er niet naar het voorkomen van een specifiek tekst in het bestand gezocht. Als het bestand voorkomt krijgt de som de status 'som gestart'. Vervolgens controleert SGWM of in hetzelfde bestand de tekst 'Hydra-NL' voorkomt en dan krijgt de som (alsnog) de status 'som gereed'.

Tot slot zijn in het MDB een drietal rekeninstellingen gedefinieerd. Allereerst een rekeninstelling voor het versienummer van Hydra-NL. Deze kan niet aangepast worden in de gebruikersschil van SGWM en daarom zetten we 'Edit' op 'false'. De tweede rekeninstelling bevat de releasedatum van Hydra-NL. Ook deze is niet aan te passen in de gebruikersschil van SGWM. De

laatste rekeninstelling bevat de naam van de gebruiker. Deze is wel te wijzigen in de gebruikersschil en daarom staat 'Edit' op 'true'. Dit ziet er dan als volgt uit in het MDB:

Definitie van rekeninstellingen in het PDB

```
<CalculationSettings>
	<CalculationSetting>
		<Name>Versienummer Hydra-NL</Name>
		<Key>Versienummer</Key>
		<Type>string</Type>
		<Value>2.4.1</Value>
		<Edit>false</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Releasedatum Hydra-NL</Name>
		<Key>Releasedatum</Key>
		<Type>string</Type>
		<Value>mei 2018</Value>
		<Edit>false</Edit>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Naam gebruiker</Name>
		<Key>Gebruikersnaam</Key>
		<Type>string</Type>
		<Value>Gebruikersnaam</Value>
		<Edit>true</Edit>
	</CalculationSetting>
</CalculationSettings>
```

Hiermee is de definitie van het PDB volledig.

De map met templatebestanden ('.../computations/templates') bevat 3 templates, waarvan het jobscript en het bsub commando script al besproken zijn. De overgebleven template betreft het invoerbestand van Hydra-NL. In dit bestand wordt veelvuldig gebruik gemaakt van diverse SGWM Keys (zowel parameters als variabelen), zoals hieronder een aantal voorbeelden (vetgedrukt) opgenomen.

Hydra-NL input-templatebestand

```
;------------algemeen------------
USERNAME                        = {Gebruikersnaam}
VERSIE                          = '{Versienummer}'
RELEASEDATUM                    = '{Releasedatum}'
DATBER                          = ''
NTHREADS                        = 4
DBRVWTYPE                       = 1
DBRVW                           =
'..\..\databases\DEMO_BenedenrivierenRijn\DEMO_BenedenrivierenRij
n.sqlite'
ODBCRVW                         =
'{Locatie}_{Type}{VolgnummerBerekeningType}'
ODBCAANMAKEN                    = JA
REGIO                           = 0
LOCATIE                         = '{Locatie}'
XCOORDINAAT                     = {Xcoordinaat}
YCOORDINAAT                     = {Ycoordinaat}
UITVOERBESTAND                  = 'uitvoer.html'
FACTOR_TP_TM                    = 1.1
PROFIEL                         =
'..\..\databases\DEMO_BenedenrivierenRijn\{Locatie}\Profielen\{Pr
ofielnaam}.prfl'
OVERZICHTNM                     = 'Overzichtsbestand_{Type}.xls'
MEMO                            = ''
RIVIER                          = 2
```

```
QMIN                            = 750
QMAX                            = 25000
QSTAP                           = 200
QAFTOP                          = 25000
MMIN                            = 0.8
MMAX                            = 7.5
MSTAP                           = 0.1
ZWS_STIJGING                    = 0
DISCRSTAPAFVMP                  = 12
UMAX                            = 50
MASTER                          = NEE
USER                            = 0
KLIMAATSCENARIO                 = 0
H_VERHOGING                     = 0
REPAREER_Q                      = JA
REPAREER_M                      = JA
;------------keringen------------
KERING                          = 1
VERDELING                       = 0
MU                              = -0.09
SIGMA                           = 0.18
ALFA                            = 0.01
;------------criterium-----------
BERTYPE                         = {TypeNR}
SEICHES                         = Nee
FREQ                            = 0.1
FREQ                            = 3.33333333333333E-02
FREQ                            = 0.01
FREQ                            = 3.33333333333333E-03
FREQ                            = 0.001
FREQ                            = 3.33333333333333E-04
FREQ                            = 0.0001
FREQ                            = 3.33333333333333E-05
FREQ                            = 0.00001
FREQ                            = 3.33333333333333E-06
FALEN                           = 1
QCR                             = {QCR}
XTRASTEUNPUNTEN                 = 0
MINOD                           = 0.0001
MAXOD                           = 0.1
NOD                             = 22
HULPDIJKENMETHODE               = 0
TABEL_FYSICA                    = NEE
TRANSPOTWIND                    =
'..\..\data\invoer\Restant\Up2U\Up2U10.dat'
PIEKPERIODE_SWAN                = JA
FACTOR_TM_TP                    = 1.1
WINDINVLOED                     = NEE
WATERSTANDSNIVEAU               = {NIVEAU}
PARAMETER_TYPE                  = {PARTYPE}
BEKLEDINGTYPE                   = '{NAAMBEKLEDING}'
REDUCTIEFACTOR                  = 1
PARAMETER_A                     = {PARAMETERA}
PARAMETER_B                     = {PARAMETERB}
PARAMETER_C                     = {PARAMETERC}
…
```

3 Stapsgewijs door SGWM

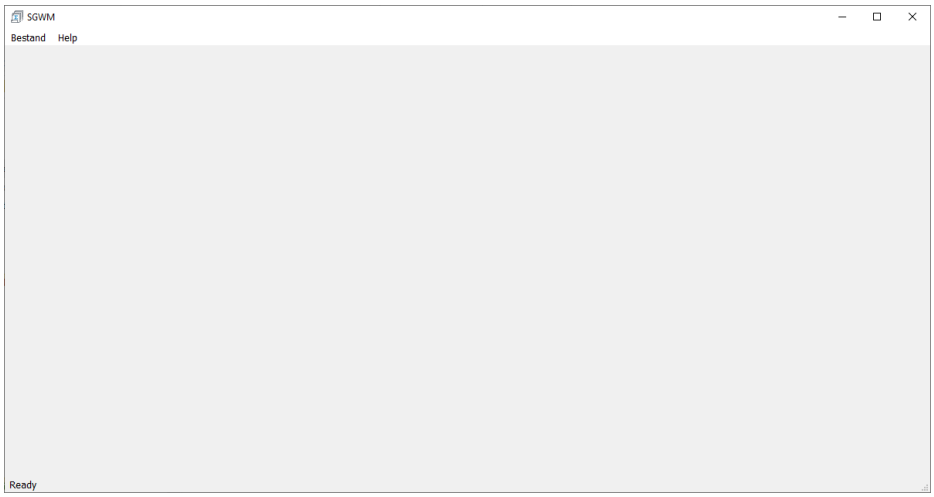
3.1 Inleiding

Dit hoofdstuk beschrijft de werking van SGWM met een focus op de bediening en onderdelen van de gebruikersschil.

3.2 Een nieuw SGWM-project aanmaken

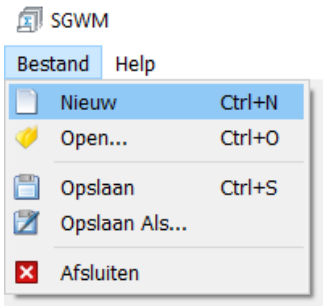
Nadat de applicatie SGWM is opgestart (door het activeren van de SGWM executable[23]) verschijnt het (lege) scherm uit Figuur 10.

Figuur 10 Leeg scherm na opstarten van SGWM



Activeer het menu 'Bestand - Nieuw' (CTRL+N) om een nieuw project te starten.

Figuur 11 Menu Bestand



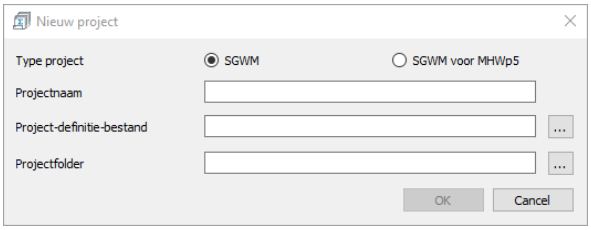
[23] Het opstarten van SGWM duurt enkele tientallen seconden in verband met uit opzetten van een tijdelijke Python omgeving waarbinnen SGWM draait.

Er verschijnt een venster waarin de instellingen van het nieuwe project gekozen moeten worden (zie Figuur 12). Allereerst moet er een keuze gemaakt worden tussen een 'gewoon' SGWM-project (met een willekeurig rekenmodel), of een SGWM-project specifiek voor MHWp5. In het geval dat voor een SGWM voor MHWp5-project wordt gekozen, wordt voor aanvullende informatie verwezen naar hoofdstuk 4. Dit hoofdstuk beschrijft het gebruik van een 'gewoon' SGWM-project voor een willekeurig rekenmodel.

Bij een nieuw SGWM-project moeten drie velden ingevuld worden:

1. De naam van het nieuwe project. Deze naam wordt ook gebruikt voor het aanmaken van een projectbestand (*.sgwm).
2. Het te gebruiken 'project-definitie-bestand' (PDB). Dit bestand (van het type XML; zie paragraaf 2.2) is het uitgangspunt van elk nieuw project en bevat onder andere verwijzingen naar (model)bestanden en de te variëren parameters van een sommenset.
3. Een folder waarin alle SGWM-projectbestanden van het nieuwe project worden aangemaakt. Kies (of maak) een folder (aan)[24] waarin alle bestanden voor het betreffende project opgeslagen gaan worden (de projectfolder).

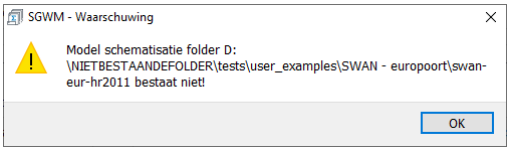
Figuur 12 Nieuw project venster



Na het invullen van een projectnaam en het selecteren van de PDB en de projectfolder, wordt een nieuw project aangemaakt door het activeren van de knop 'OK'.

Bij het aanmaken van elk nieuw project wordt de inhoud van het PDB ingelezen en gecontroleerd[25]. Wanneer fouten voorkomen in dit bestand, wordt dat teruggekoppeld met een foutmelding op het scherm. In de foutmelding uit Figuur 13 klopt bijvoorbeeld de verwijzing van het pad met de data van de modelschematisatie niet.

Figuur 13 Foutmelding in PDB

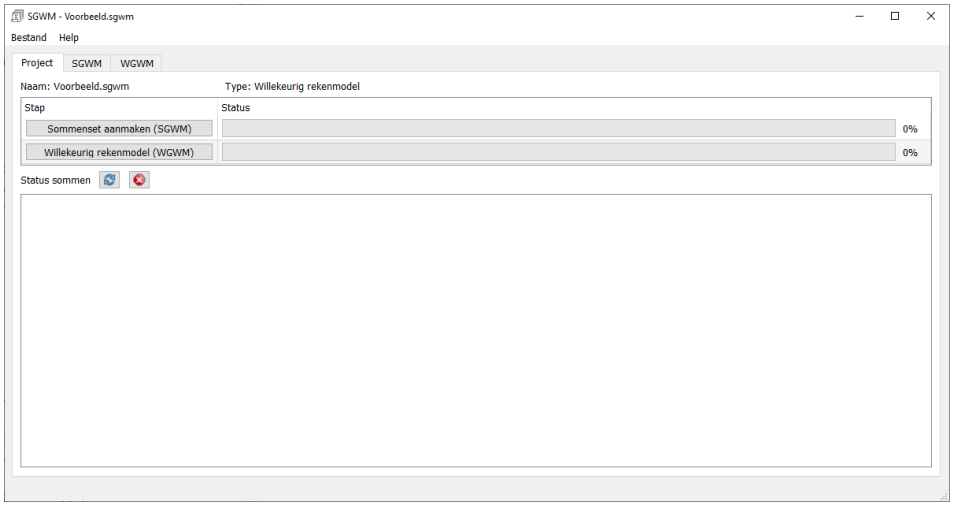


[24] De keuze voor een bestandstructuur is helemaal vrij. In de voorbeelden hebben we in de map van de modelschematisatie een subfolder 'sgwm' aangemaakt waarin het projectbestand, maar ook het PDB worden opgeslagen.
[25] Deze controle wordt uitgevoerd door onder andere de XML te verifiëren tegen een XSD-bestand (zie ook paragraaf 2.2 en bijlage G).

Wanneer er geen fouten geconstateerd worden in het PDB wordt het project geladen en verschijnt de interface van SGWM met drie tabbladen:

1. Een tabblad 'Project', met daarin de status van de sommen in het project. Bij een nieuw aangemaakt project is de tabel met sommen initieel nog leeg en hebben de voortgangsindicatoren nog geen status waarde.
2. Een tabblad 'SGWM', waarin een definitie van de SGWM sommenset kan worden gemaakt.
3. Een tabblad 'WGWM[26]', waarin een overzicht van alle sommen gegeven wordt en waarin voor (een selectie van) de sommen invoerbestanden kunnen worden gegenereerd.

Figuur 14 Gebruikersschil met drie tabbladen



Na het aanmaken van een nieuw project zijn er twee nieuwe bestanden opgeslagen in de gekozen projectfolder:

- Een projectbestand (van het type SGWM; zie Bijlage A);
- Een logbestand van SGWM (van het type log; zie Bijlage E).

3.3 Een bestaand project openen

Via het menu 'open..' (zie Figuur 11) kan een bestand SGWM worden geopend. Er verschijnt een bestandsdialoog waarin een bestaand SGWM-projectbestand (zie Bijlage A) geselecteerd kan worden. Bij het openen van het project worden alle project bestanden, zoals gedefinieerd in het SGWM-projectbestand geopend, waaronder ook het PDB en eventueel het MDB. Als daarbij geen fouten worden geconstateerd dan verschijnt het scherm uit Figuur 14.

3.4 Een sommenset definiëren

Een sommenset in SGWM bestaat uit de definitie van alle mogelijke combinaties van parameterwaarden (bij gebruik van WAQUA, SWAN of D-Hydro zijn

[26] WGWM staat voor Willekeurige Generator WaterModellen en is daarmee een variant van de modules zoals deze in SGWM voor MHWp5 zijn benoemd, zoals RGWM (Randvoorwaarden Generator WaterModellen), BGWM (Beginvoorwaarden Generator WaterModellen), GGWM (Golf Generator WaterModellen), etc.

de parameters de stochasten of randvoorwaarden; meestal aangeduid met belastingcombinaties).

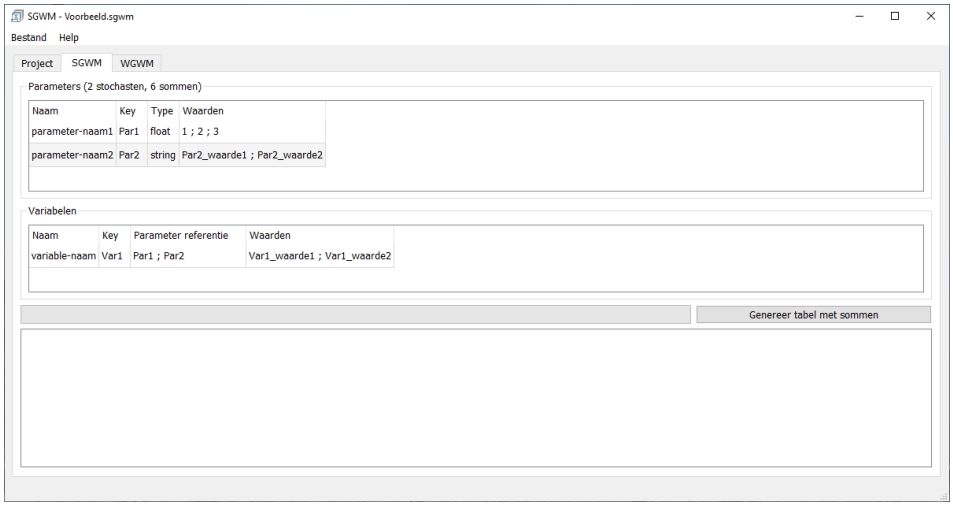
Stel dat een WAQUA-som gemaakt moet worden met twee stochasten: (1) de rivierafvoer en (2) het meerpeil. In de sommenset wordt dan gedefinieerd welke waarden beide stochasten moeten kunnen aannemen in de sommen die gegenereerd gaan worden. Stel dat voor beide stochasten drie mogelijke waarden gedefinieerd zijn, te weten een rivierafvoer gelijk aan: 100, 200 en 300 m^3/s en een meerpeil gelijk aan: 1, 3 en 5 m+NAP. Dan bestaat de sommenset uit drie maal drie, negen combinaties. Er zullen dan negen sommen met SGWM gegenereerd kunnen worden, namelijk:

- Som 1: rivierafvoer 100 m^3/s en meerpeil 1 m+NAP.
- Som 2: rivierafvoer 100 m^3/s en meerpeil 3 m+NAP.
- Som 3: rivierafvoer 100 m^3/s en meerpeil 5 m+NAP.
- Som 4: rivierafvoer 200 m^3/s en meerpeil 1 m+NAP.
- Som 5: rivierafvoer 200 m^3/s en meerpeil 3 m+NAP.
- Som 6: rivierafvoer 200 m^3/s en meerpeil 5 m+NAP.
- Som 7: rivierafvoer 300 m^3/s en meerpeil 1 m+NAP.
- Som 8: rivierafvoer 300 m^3/s en meerpeil 3 m+NAP.
- Som 9: rivierafvoer 300 m^3/s en meerpeil 5 m+NAP.

De sommenset wordt altijd bepaald uit de unieke set van alle mogelijke combinaties van parameterwaarden (belastingcombinaties of randvoorwaarden).

Bij het aanmaken van een nieuw project of het laden van een bestaand project in SGWM toont de tabel 'Parameters' in het tabblad 'SGWM' alle parameters met alle mogelijke waarden, zoals gedefinieerd in het PDB. Voor elke parameter worden in dit scherm de naam, de zogenoemde SGWM Key, die de parameter binnen SGWM duidt, en de mogelijke waarden per parameter getoond (zie Figuur 15).

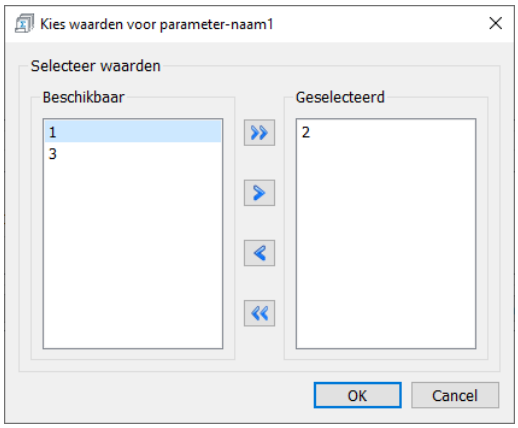
Figuur 15
Definitie sommenset



In de titel van het parameteroverzicht staat ook het aantal sommen dat op basis van de unieke combinatie van parameters door SGWM aangemaakt wordt als de sommenset wordt gegenereerd.

Voor elke parameter worden initieel alle mogelijke waarden van de betreffende parameter uit het PDB getoond. Het is mogelijk om voor het project een sommenset te definiëren met minder waarden (resulterend in minder sommen). Door op een veld met waarden van een parameter te klikken, wordt een popup scherm geactiveerd zoals weergegeven in Figuur 16.

Figuur 16 Selecteer één of meerdere waarden per parameter



In het overzicht aan de rechterzijde staan alle waarden, die geselecteerd zijn voor de definitie van de sommenset. In het overzicht aan de linkerzijde staan alle waarden die niet geselecteerd zijn voor de definitie van de sommenset maar wel beschikbaar zijn vanuit het PDB en dus eventueel geselecteerd kunnen worden voor de definitie van de sommenset. Met de knoppen tussen de overzichten kunnen één of meerdere parameters van het rechter naar het linker overzicht of andersom verplaatst worden.


Met deze knop worden alle waarden geselecteerd en in het rechter overzicht geplaatst.


Met deze knop wordt een enkel geselecteerde waarde in het linker overzicht, naar het rechter overzicht verplaatst.


Met deze knop wordt een enkel geselecteerde waarde in het rechter overzicht, naar het linker overzicht verplaatst.


Met deze knop worden alle waarden in het linker overzicht geplaatst.

Er moet minimaal één waarde aanwezig zijn in het rechter venster (‘Geselecteerd’) aanwezig zijn om de selectie te kunnen bevestigen met de knop ‘OK’.

Het totaal aantal sommen dat in de titel van het parameteroverzicht weergegeven wordt, wordt bijgewerkt zodra voor één van de parameters, waarden worden aangepast.

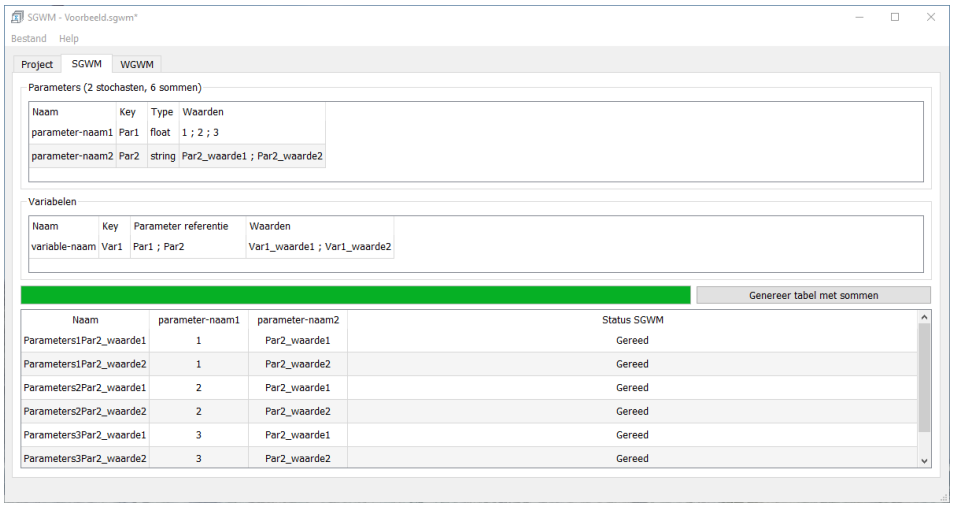
Onder de tabel met parameters wordt de tabel met variabelen getoond. Deze tabel bevat de naam van de variabele, de SGWM Key, waarmee de variabele binnen SGWM wordt geduid, de Key van de parameter, waarvan de variabele afhankelijk is gemaakt en de mogelijke waarden per variabele (zie Figuur 15). Zodra de waarden van een parameter worden aangepast (zoals

hiervoor beschreven) dan wordt ook het aantal waarden van de variabele die aan deze parameter is gekoppeld, aangepast.

Nadat de keuze voor de sommenset definitief is, kan de tabel met sommen aangemaakt worden door de knop 'Genereer tabel met sommen' te activeren. De tabel onder in het scherm 'SGWM' wordt gevuld met alle sommen uit de sommenset, zoals gedefinieerd door de selectie van parameters. Elke regel in de tabel correspondeert met één som. Voor elke som worden kenmerken van de som weergegeven in kolommen. Dit zijn achtereenvolgens van links naar rechts:

a. De naam van elke som, de zogenoemde (uniek) somID.
b. De waarde van elke parameter in de betreffende som (dit kunnen één of meerdere kolommen zijn afhankelijk van het aantal parameters in de definitie van een PDB).
c. De status van het aanmaken van een som in de sommenset. N.B. deze status is altijd voor elke som gelijk aan 'Gereed' na het genereren van de tabel met sommen.

Figuur 17
Gevulde tabel met een sommenset op het tabblad 'SGWM'

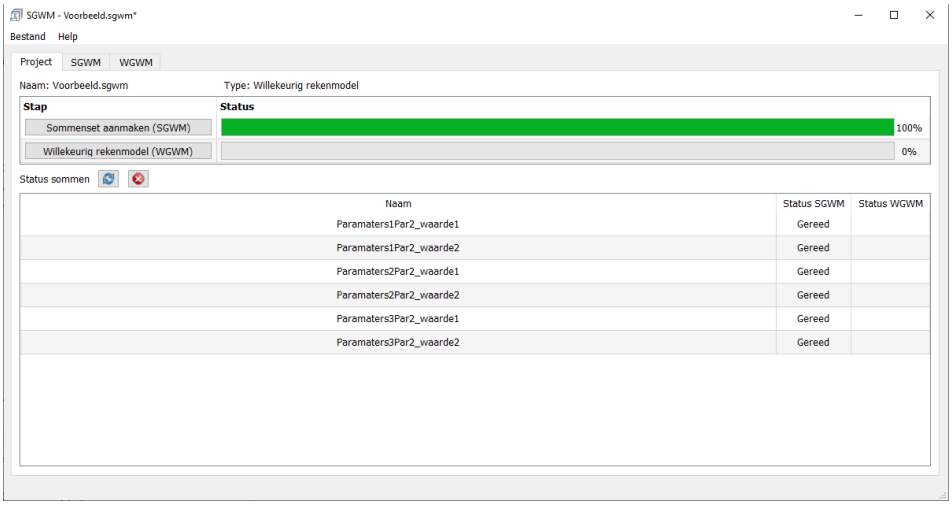


Mappen voor elke som

Bij het genereren van de tabel met sommen wordt ook voor elke som een map aangemaakt in de 'Computation' folder (zoals gedefinieerd in het PDB. Het kan voorkomen dat de 'Computation' folder al mappen en bestanden bevat van een voorgaand project. Er wordt daarom een waarschuwing getoond dat deze folder door SGWM automatisch leeg wordt gemaakt bij het genereren van de sommen. Maak eventueel een kopie van een al bestaande 'Computation' folder om te voorkomen dat invoer en mogelijk ook uitvoer bestanden van bestaande sommen verloren gaan!

Na het genereren van de sommenset wordt ook de tabel met sommen in het tabblad 'Project' gevuld met alle sommen. Het tabblad 'Project' is bedoeld om de status van de verschillende stappen uit het SGWM-project te monitoren. Dit wordt met name relevant in projecten van het type SGWM voor MHWp5, omdat daar meerdere modellen in één SGWM-project worden beschouwd.

Figuur 18
Tabel met een sommenset op het tabblad 'Project'



In dit tabblad zijn twee voortgangsbalken opgenomen, die aangeven welk deel van de sommenset gereed is. In Figuur 18 is te zien dat alle sommen zijn aangemaakt (voortgang Sommenset aanmaken is 100%), maar dat er nog geen resultaten zijn van de sommen (berekend met een willekeurig rekenmodel[27]). De tabel onder de voortgangsbalken toont de gehele sommenset en geeft in twee kolommen de status per som weer voor respectievelijk 'het aanmaken van de sommenset' en 'het resultaat van de som met een willekeurig rekenmodel'.

De definitie van de sommenset wordt weggeschreven in het bestand 'sommenset.xml' (in de projectfolder) als het project wordt opgeslagen. Zie voor een beschrijving van dit bestand Bijlage B. Daarnaast wordt een CSV-bestand aangemaakt waarin alle sommen uit de sommenset zijn opgenomen. Dit bestand ('sommen.csv') wordt eveneens in de projectfolder weggeschreven. Zie voor een beschrijving van dit bestand Bijlage C.

Aangepaste sommenset
Het kan voorkomen dat het ongewenst is om alle combinaties van parameterwaarden onderdeel te laten zijn van de sommenset. Bepaalde combinaties kunnen op voorhand niet interessant zijn; bijvoorbeeld een combinatie van hoge zeewaterstanden met een oostelijke windrichting. Het is dan zinvol om deze combinaties uit de sommenset te verwijderen. Dit kan niet via de gebruikersinterface van SGWM. Deze maakt immers altijd de volledige set aan op basis van alle mogelijk combinaties van geselecteerde parameterwaarden. Daarom zal een handmatige aanpassing gedaan moeten worden in het CSV-bestand met sommen (sommen.csv). Open dit bestand in een tekst-editor of in MS Excel en verwijder de regels van de sommen die niet interessant zijn. Sla de wijzigingen op in het CSV-bestand. Open het project opnieuw in de SGWM-applicatie (via het menu 'Bestand – Open…').

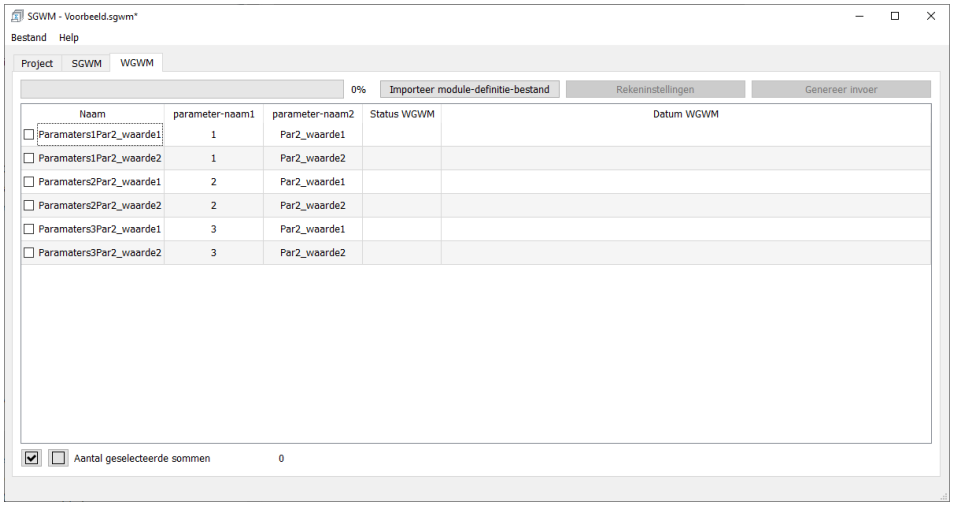
[27] De status van een som wordt bepaald op basis van de definitie van 'States' in het MDB (zie paragraaf 2.3.6). Hierin wordt ook aangegeven op basis van welke 'State' de voortgang van de sommen moet worden gemeten.

3.5 Invoer genereren

In het tabblad ‘WGWM’ kan de invoer voor één of meerdere som worden aangemaakt. Ook dit tabblad bevat een overzicht van alle sommen in de sommenset (zie ook hoofdstuk 3.4). Elke regel in de tabel correspondeert met één som. De kolommen bevatten achtereenvolgens van links naar rechts:

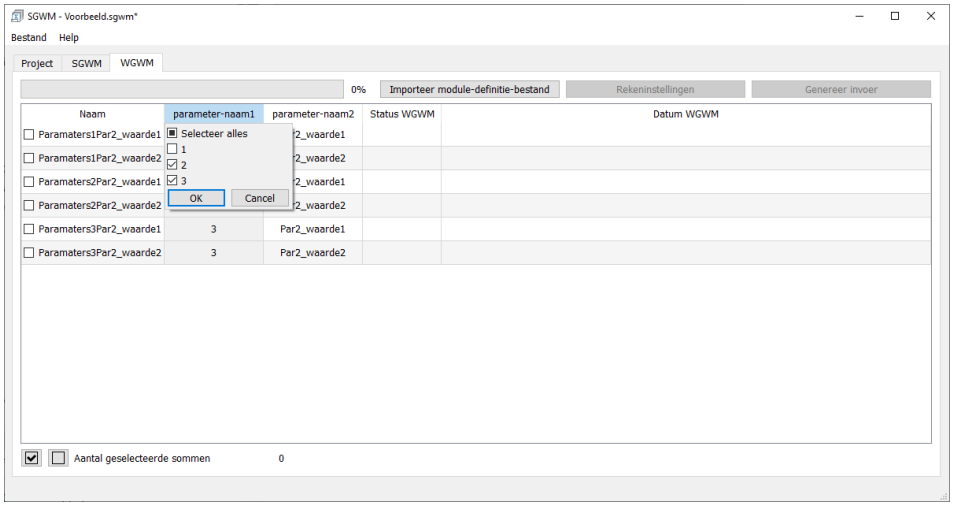
a. Een vinkveld waarmee een som in deze tabel kan worden toegevoegd aan een selectie.
b. De unieke somID zoals gedefinieerd in het PDB.
c. De waarde van elke parameter in de betreffende som (dit kunnen één of meerdere kolommen zijn afhankelijk van het aantal parameters in de definitie van een PDB).
d. De status van de som.
e. De datum en tijd waarop de laatste status verandering heeft plaatsgevonden.

Figuur 19 Overzicht met sommen in het tabblad ‘WGWM’



De kolommen met parameterwaarden (zie punt c) en de kolom met de status van de sommen kunnen gefilterd worden, door op de kolomtitel te klikken. Er verschijnt een filter popup.

Figuur 20 Filteropties



De tabel met sommen bevat alle sommen uit een sommenset. Het is onwaarschijnlijk dat alle sommen in één batch tegelijk aan een rekengrid aangeboden zullen worden. Vaak zal eerst een beperkte set sommen uitgevoerd worden ter beoordeling van de resultaten op basis van de gemaakte keuze in de modelschematisatie en instellingen van de software. Ook daarna is het waarschijnlijk dat sommen in subsets van de totale sommenset uitgevoerd zullen worden, als is het maar om overzicht op de voortgang te behouden. SGWM bevat functionaliteit om dit proces te ondersteunen.

Stap 1 is het selecteren van sommen uit de totale sommenset voor de definitie van een rekenbatch (een subset). Sommen kunnen aan een subset toegevoegd worden door vinkjes te plaatsen in de eerste kolom van de tabel. Het aantal geselecteerde sommen (in de subset) wordt onder de tabel weergegeven.

Gebruik eventueel (slimme) filtercombinaties om de tabel te verkleinen tot een set sommen die aan een subset toegevoegd moeten worden, bijvoorbeeld een subset met alle sommen van een afvoer gelijk aan een specifieke waarde of een specifiek meerpeil.

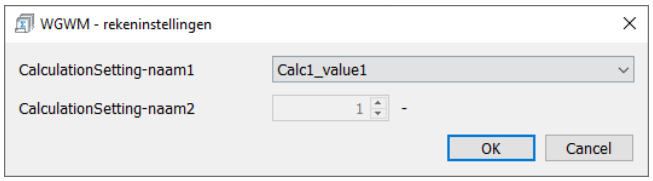
Gebruik de knoppen ☑ en ☐ om vervolgens in één handeling alle sommen dan wel toe te voegen aan de subset, respectievelijk te verwijderen uit de subset.

Let op: het aantal geselecteerde sommen dat onder de tabel wordt weergegeven is gebaseerd op het aantal aangevinkte sommen in de set sommen die **zichtbaar zijn in de tabel.** Het is ook dit aantal, waarvan de invoer wordt gemaakt als er op 'genereer invoer' wordt gedrukt. Een filter kan het aantal geselecteerde sommen dus beïnvloeden als deze, als gevolg van een filter, onzichtbaar worden in de tabel. Als je het totaal aantal sommen dat aangevinkt is wilt zien en/of van alle aangevinkte (geselecteerde) invoer de invoer wilt genereren, moeten alle filterselecties opgeheven worden (kies voor elke kolom 'selecteer alles' in de het filter menu).

Stap 2 is het inlezen van de specifieke rekenmodel definities uit het MDB (zie paragraaf 2.3). Door de knop 'Importeer module-definitie-bestand' te activeren wordt er een bestandsdialoog getoond waarin het MDB kan worden geselecteerd. Na het selecteren van dit bestand wordt de inhoud van het bestand gecontroleerd aan de hand van een XSD-bestand (zie ook bijlage G en paragraaf 2.3). Wanneer het MDB zonder fouten is ingelezen wordt de knop 'Rekeninstellingen' actief.

Stap 3 is het eenmalig definiëren van de rekeninstellingen voor de sommen. Deze rekeninstellingen kunnen een standaardwaarde hebben gekregen vanuit het MDB (zie ook paragraaf 2.3.7). Het scherm met rekeninstellingen moet minimaal één keer geopend zijn, voordat sommen gegenereerd kunnen worden. Open het scherm met rekeninstellingen door de knop 'Rekeninstellingen' te activeren. Er verschijnt een popup venster met rekeninstellingen zoals weergegeven in Figuur 21.

Figuur 21
Rekeninstellingen



De rekeninstellingen die weergegeven worden in dit scherm zijn afhankelijk van de definitie van de zogenoemde 'CalculationSettings' in het MDB en zijn vaak specifiek (configureerbaar) voor een rekenomgeving (bijvoorbeeld het rekengrid bij SSC-Campus of een eigen rekencluster).

Bij het sluiten van het scherm met rekeninstellingen (door de knop 'OK' te activeren), worden de gekozen rekeninstellingen naar het bestand 'rekeninstellingen.xml' in de projectfolder weggeschreven. Zie voor een beschrijving van dit bestand Bijlage D.

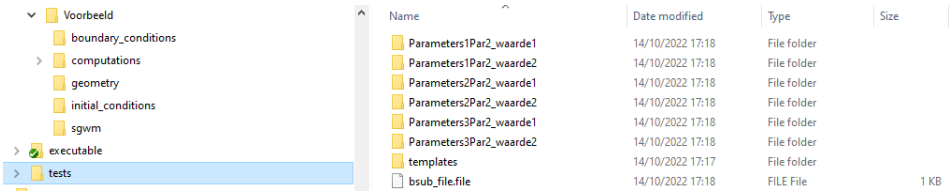
Stap 4 is het genereren van de invoerbestanden voor de verschillende sommen uit de gedefinieerde subset of de gehele tabel met sommen. Activeer de knop 'Genereer invoer' om invoerbestanden voor elke som aan te maken inclusief een job commando script voor het starten van de sommen. De voortgang van het wegschrijven van de invoerbestanden per som, wordt weergegeven met een voortgangsbalk boven in het scherm.

Let op: het aantal sommen waarvoor invoer weggeschreven wordt is gelijk aan het aantal aangevinkte sommen in **de zichtbare set sommen in de tabel**. Een filter kan het aantal geselecteerde sommen dus beïnvloeden als deze, als gevolg van een filter, onzichtbaar zijn in de tabel. Het aantal sommen waarvoor invoer weggeschreven wordt, is altijd gelijk aan het aantal sommen dat onder de tabel met sommen wordt weergegeven.
Voor elk som wordt in de folder '…/computations/'[28] een folder aangemaakt met de foldernaam gelijk aan de somID[29]. In elke folder bevinden zich de invoerbestanden voor de betreffende som. Afhankelijk van de modelsoftware die gebruikt wordt (WAQUA, SWAN, D-Hydro, Hydra-NL, etc.), kunnen de bestanden verschillen (zoals gedefinieerd onder de templates in het MDB, zie paragraaf 2.3.4). Minimaal zal er een input-templatebestand neergezet worden en een lsf-job-script.

Daarnaast wordt er batch bestand aangemaakt in de folder '…/computations/'[28] waarin het BSUB-commando script is opgenomen voor de sommen waarvan de invoer is weggeschreven. De naamgeving van dit bestand is afhankelijk van de definitie in het PDB.

[28] Of een alternatieve foldernaam zoals gedefinieerd is in het PDB.
[29] De Id per som zoals gedefinieerd is in het PDB eventueel aangevuld met een subfolder zoals gedefinieerd in het MDB.

Figuur 22 Invoerfolder per som en batch-bestand met BSUB commando script



Let op dat wanneer een tweede subset met sommen wordt aangemaakt het batch-bestand met BSUB-commando script (met dezelfde naam) wordt overschreven. Aanbevolen wordt daarom het batch bestand met BSUB-commando script standaard te hernoemen zodra deze is aangemaakt.

3.6 Controle invoer

In de projectfolder wordt het SGWM-logbestand (zie Bijlage E) elke keer bijgewerkt bij het aanmaken van invoer voor sommen. Dit logbestand bevat uitgebreide informatie over het aanmaken van de invoer voor de verschillende sommen. Gebruik dit bestand om te controleren of er geen fouten zijn gemaakt in de definitie van de sommen. In de template-bestanden (bijvoorbeeld een siminp-bestand voor WAQUA) kunnen verwijzingen opgenomen zijn naar invoerbestanden in de modelschematisatie (folders). SGWM controleert bij het inlezen van de template-bestanden of de bestanden waarnaar verwezen wordt, onderdeel uitmaken van de modelschematisatie en dat de verwijzing naar het bestand klopt. Als dat niet zo is dan wordt dit gelogd met een label [ERROR]. Zie Bijlage E voor een voorbeeld van een logbestand.

3.7 Sommen uitvoeren

In deze versie van SGWM is het nog niet mogelijk om sommen direct vanuit de gebruikersschil te starten. Mogelijk wordt dit in de toekomst wel gerealiseerd. Vooralsnog moet een gebruiker het submit-script 'handmatig' via bijvoorbeeld Putty naar LSF versturen. LSF handelt vervolgens de rekenopdrachten af. Zie Bijlage G voor een overzicht van een aantal veel gebruikte Putty commando's.

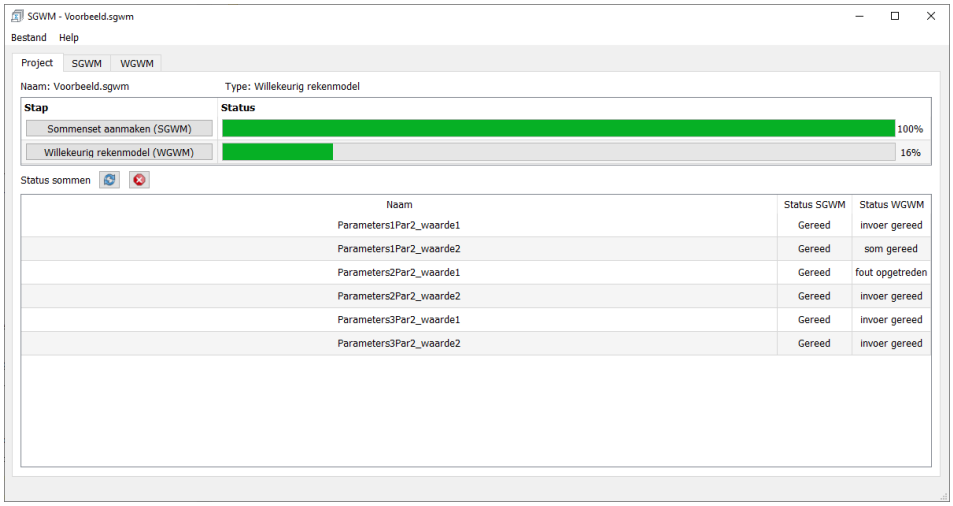
3.8 Status sommen bijwerken

Elke som uit de sommenset in SGWM krijgt meerdere statussen. Er wordt in SGWM zowel een statusvoortgang geven van het aanmaken van de sommenset als van de sommen zelf (denk hierbij aan het genereren van de invoerbestanden en/of het beschikbaar hebben van resultaten van de sommen). Een compleet overzicht van de status van de sommen in SGWM wordt weergegeven in het tabblad 'Project'. In Figuur 23 is een voorbeeld weergegeven. In de voortgangsbalken is te zien dat de gehele sommenset

voor dit project is aangemaakt (voortgangsindicator 'Sommenset aanmaken' is gelijk aan 100%) en dat 1/6 van de sommen gereed is en succesvol uitgevoerd[30] (voortgangsindicator 'Willekeurige rekenmodel is gelijk aan 16%)'.

Om de status informatie bij te werken dient de ververs knop  achter het label 'Status sommen' geactiveerd te worden. Zodra deze knop wordt geactiveerd controleert SGWM alle mappen in de folder '…/computations/'[28] op het voorkomen van specifieke bestanden met eventueel specifieke teksten die aan een bepaalde statusdefinitie zijn gekoppeld (zie ook paragraaf 2.3.6). In sommige projecten is het aantal sommen zo groot dat het bepalen van de status van de sommen aanzienlijke tijd vergt. Het is daarom mogelijk om deze actie te onderbreken met de cancel knop  . Daarnaast zal SGWM de status van een specifieke som niet meer controleren zodra deze de status heeft verkregen die gekoppeld is aan de voortgangsindicator[30]. Dit maakt dat naarmate de voortgang toeneemt, de tijd voor het bepalen van de status (van sommen die nog niet gereed zijn) steeds korter wordt.

Figuur 23 Status van sommen in het tabblad 'Project'



3.9 Opslaan van een project

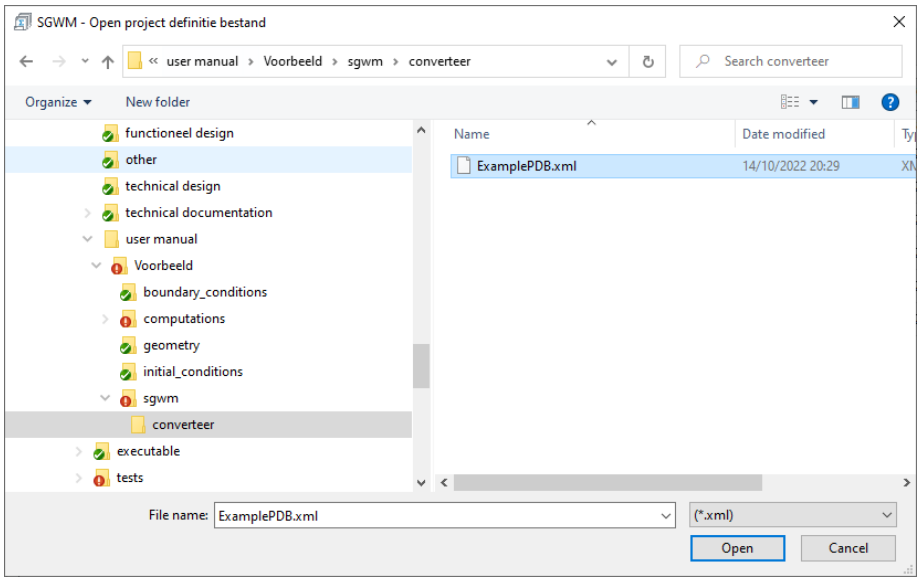
Het project kan op elk moment opgeslagen worden door de menu-optie 'Bestand - Opslaan' (CTRL+S) te activeren. Een kopie van een project kan aangemaakt worden door de menu-optie 'Bestand – Opslaan als' te kiezen. Alle wijzingen worden opgeslagen in de verschillende bestanden in de SGWM-projectfolder.

[30] Vanaf versie SGWM 2.0.0 kan een gebruiker zelf de statussen van de sommen definiëren in het MDB (zie hiervoor ook paragraaf 2.3.6). Minimaal 1 van deze statusdefinities moet gekoppeld worden aan de voortgangsbalk via de XML-tag <Progress> (gelijk aan true). De voortgang van de sommen wordt dan aan de hand van deze status bepaald.

3.10 Converteer een oud PDB

Vanaf versie 2.1.0 van SGWM is de optie ‘Converteer pdb…’ opgenomen om een project-definitie-bestand van een bestaand project (van voor versie 2.1.0) te converteren naar het nieuwe formaat dat vereist is voor deze versie (met name de nieuwe definitie van de variabelen; zie de release notes van SGWM). Selecteer een project-definitie-bestand om te converteren en open dit bestand. De conversie wordt meteen gestart en het nieuwe bestand komt in dezelfde map te staan en heeft in de dezelfde naam maar met de toevoeging “_new”.

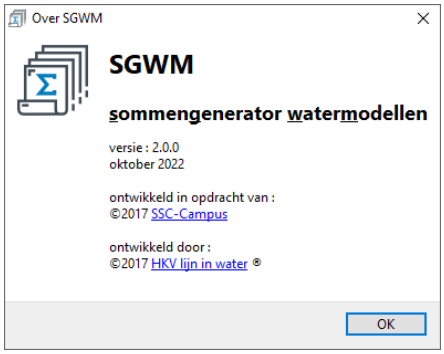
Figuur 24
Converteer een PDB



3.11 Help (over SGWM)

Het scherm uit Figuur 25 kan opgeroepen worden via de menu-optie ‘Help – Over…’ (F1). Het scherm geeft informatie over SGWM en de versie die gebruikt wordt.

Figuur 25
Over SGWM



3.12 Afsluiten

De applicatie SGWM kan afgesloten worden via de menu-optie ‘Bestand - Afsluiten’. Als er wijzigingen zijn gemaakt wordt gevraagd of deze wijzigingen opgeslagen moeten worden voordat SGWM afgesloten wordt.
Let op: bij het afsluiten van de applicatie met het kruisje in de rechterbovenhoek van SGWM wordt deze controle niet uitgevoerd! Sluit SGWM dus altijd af via het menu!

4 SGWM voor MHWp5

4.1 Inleiding

Dit hoofdstuk beschrijft aanvullend op de beschrijving uit hoofdstuk 3, alle verschillen en/of aanvullingen in het geval dat in SGWM gekozen wordt voor een project van het type 'SGWM voor MHWp5'. Dit specifieke type project is toegevoegd aan SGWM om het gebruik van een modellentrein uit de Maatgevend Hoogwater processor te ondersteunen. Tot SGWM versie 2.0.0 was er een aparte SGWM applicatie voor MHWp5 met een eigen handleiding (Thonus, 2019). Vanaf SGWM versie 2.0.0 is deze versie volledig geïntegreerd met SGWM, met als voornaamste voordeel, dat er in plaats van twee, slechts één broncode beheerd hoeft te worden. Deze integratie heeft ertoe geleid dat er een aantal aanpassingen zijn doorgevoerd in het gebruik van SGWM voor willekeurige rekenmodellen (SGWM versies voor versie 2.0.0; zie hiervoor de release notes van SGWM) en aanpassingen voor SGWM specifiek voor MHWp5. De aparte handleiding van SGWM voor MHWp5 is nu ook geïntegreerd in deze handleiding, waarmee (Thonus, 2019) vervalt.

4.2 MHWp5

De Maatgevend Hoogwater processor versie 5 (afgekort met MHWp5) is een toepassing die bestaat uit een verzameling 'afzonderlijke' rekenmodules met als primair doel het inschatten van de effecten van variaties in randvoorwaarden en/of faalmodi en sluitregimes, op de hoogwaterveiligheid van een beschouwd gebied. Elke module vervult een specifiek onderdeel (functie) uit een modellentrein om dit doel te bereiken.

De rekenmodules uit de modellentrein kunnen in drie clusters opgedeeld worden:

1. Rekenmodules voor berekeningen met een waterbewegingsmodel en golfrandvoorwaarden, voor verschillende combinaties van belastingstochasten.
2. Rekenmodules voor het maken van een probabilistische database.
3. Rekenmodules voor het maken van probabilistische berekeningen met een Hydra-model voor verschillende locaties in het beschouwde gebied.

De verschillende modules worden 'aangestuurd' door één gemeenschappelijke module, SGWM (voor MHWp5). SGWM heeft ook hier als primaire taak het genereren van invoerbestanden (op basis van template-bestanden) voor elke unieke som in de verschillende modules. De verzameling modules tezamen wordt aangeduid met MHWp5.

4.3 Stapsgewijs door SGWM

Bij het aanmaken van een nieuw SGWM-project verschijnt er een venster waarin de instellingen van het nieuwe project gekozen moeten worden (zie Figuur 26Figuur 12). Als de keuze gemaakt wordt voor een SGWM-project specifiek voor MHWp5 verschijnen er vier velden die ingevuld moeten worden:

1. Kies een type MHWp5 project. Er zijn inmiddels 4 mogelijkheden:
   a. een project waarin een sommenset aangemaakt moet worden voor begincondities;
   b. een project waarin een sommenset aangemaakt moet worden waterbewegingsberekeningen;
   c. een project waarin een sommenset aangemaakt moet worden voor het genereren van een probabilistische database;
   d. een project waarin een sommenset aangemaakt moet worden voor probabilistische berekeningen;

   In de projecten voor de begincondities en waterbewegingsberekeningen worden sommen gedefinieerd op basis van verschillende combinaties van belastingen en faalmodi, bijvoorbeeld verschillende afvoeren, zeewaterstanden en faalmodi van de Maeslantkering. In het project voor het genereren van een probabilistische database worden sommen aangemaakt op basis een type Hydra-database en een riviertraject[31]. Tot slot worden in het project voor de probabilistische berekeningen sommen gedefinieerd op basis van uitvoerlocaties en Hydra-instellingen.
2. De naam van het nieuwe project. Deze naam wordt ook gebruikt voor het aanmaken van een projectbestand (*.sgwm).
3. Het te gebruiken 'project-definitie-bestand' (PDB). Dit bestand (van het type XML; zie paragraaf 2.2) is het uitgangspunt van elk nieuw project en bevat onder andere verwijzingen naar (model)bestanden en de te variëren parameters van een sommenset.
4. Een folder waarin alle SGWM-projectbestanden van het nieuwe project worden aangemaakt. Kies (of maak) een folder (aan)[32] waarin alle bestanden voor het betreffende project opgeslagen gaan worden (de projectfolder).

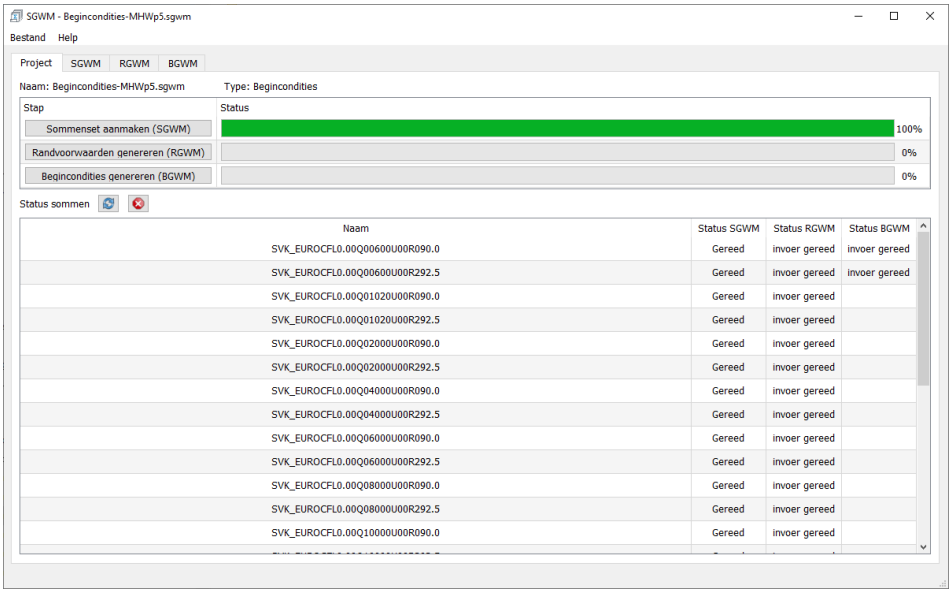
Figuur 26 Nieuw project venster

[31] Er wordt met riviertrajecten gewerkt om de omvang van de database te beperken.
[32] De keuze voor een bestandstructuur is helemaal vrij. In de voorbeelden hebben we in de map van de modelschematisatie een subfolder 'sgwm' aangemaakt waarin het projectbestand, maar ook het PDB worden opgeslagen.

Na kiezen van een type MHWp5 project, het invullen van een projectnaam en het selecteren van de PDB en de projectfolder, wordt een nieuw project aangemaakt door het activeren van de knop 'OK'.
Afhankelijk van het type MHWp5-project verschijnen er verschillende tabbladen in SGWM.

Voor een Begincondities MHWp5 project zijn dat de tabbladen 'SGWM', 'RGWM' en 'BGWM'. De functionaliteit van het tabblad 'SGWM' is identiek aan SGWM voor een willekeurig rekenmodel en de tabbladen 'RGWM' en 'BGWM' zijn gelijk van opzet aan dat van 'WGWM' in SGWM voor een willekeurig rekenmodel. In het tabblad 'RGWM' worden invoerbestanden aangemaakt waarmee met RGWM (de Randvoorwaarden Generator WaterModellen) randvoorwaarden-bestanden gegenereerd worden die nodig zijn voor het uitvoeren van de waterbewegingsberekeningen (en die de begincondities moeten gaan vormen voor de definitieve waterbewegingsberekeningen). In het tabblad 'BGMW' worden invoerbestanden aangemaakt voor de SingleRunner of en soortgelijk waterbewegingsmodel (bijvoorbeeld Sobek of D-Hydro) waarmee de waterbewegingsberekeningen worden uitgevoerd die als begincondities gebruikt gaan worden.

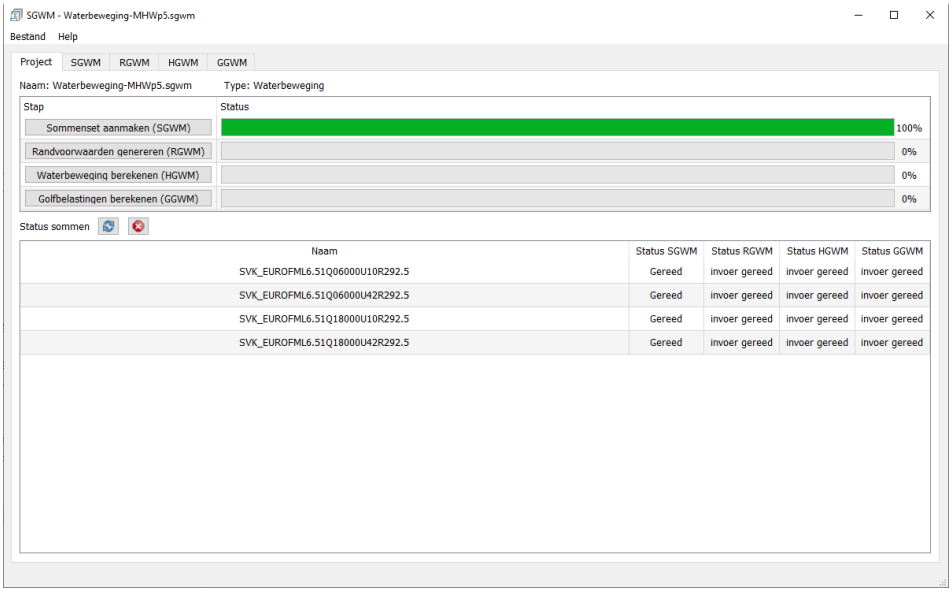
Figuur 27 Tabblad in een Begincondities MHWp5 project



Voor een Waterbeweging MHWp5 project zijn dat de tabbladen 'SGWM' en 'RGWM', 'HGWM' en 'GGWM'. De functionaliteit van het tabblad 'SGWM' is identiek aan SGWM voor een willekeurig rekenmodel en van de overige tabbladen gelijk van opzet aan dat van 'WGWM' in SGWM voor een willekeurig rekenmodel. In het tabblad 'RGWM' worden invoerbestanden aangemaakt waarmee met RGWM (de Randvoorwaarden Generator WaterModellen) randvoorwaarden-bestanden gegenereerd worden die nodig zijn voor het uitvoeren van de waterbewegingsberekeningen. In het tabblad 'HGMW' worden invoerbestanden aangemaakt voor de SingleRunner of en soortgelijk waterbewegingsmodel (bijvoorbeeld Sobek of D-Hydro) waarmee de waterbewegingsberekeningen worden uitgevoerd. In het tabblad 'GGWM' worden invoerbestanden aangemaakt voor de GolfGenerator Watermodellen

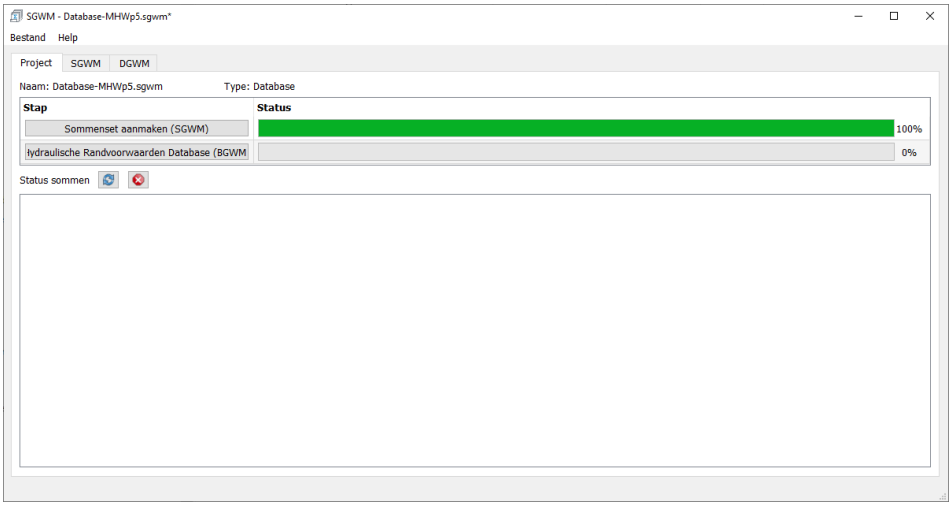
waarmee golfcondities bij worden berekend voor een van tevoren vastgestelde set uitvoerlocaties.

Figuur 28 Tabblad in een Waterbeweging MHWp5 project



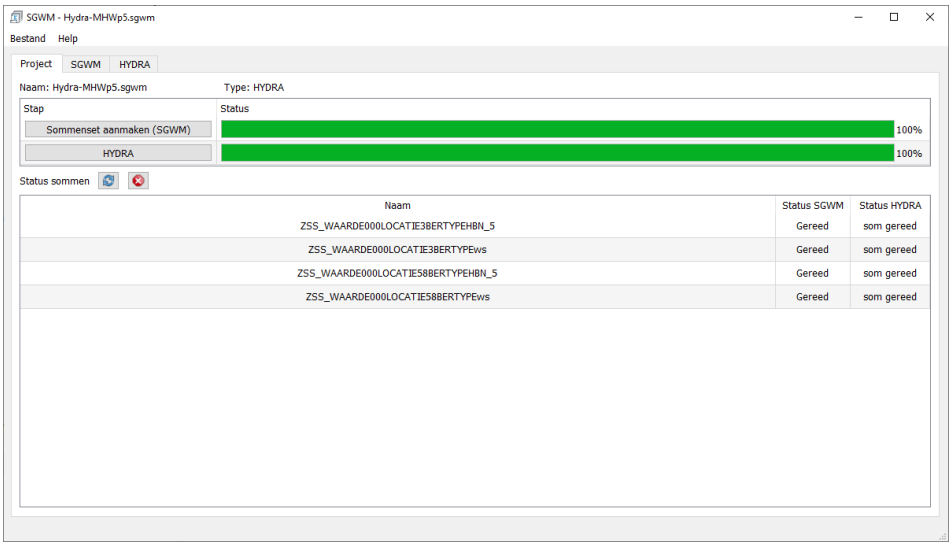
Voor een Database MHWp5 project zijn dat de tabbladen ‘SGWM’ en ‘DGWM. De functionaliteit van het tabblad ‘SGWM’ is identiek aan SGWM voor een willekeurig rekenmodel en van de overige tabbladen gelijk van opzet aan dat van ‘WGWM’ in SGWM voor een willekeurig rekenmodel. In het tabblad ‘DGWM’ worden invoerbestanden aangemaakt waarmee met Database generator hydraulische databases gegenereerd, die nodig zijn voor het uitvoeren van de probabilistische berekeningen.

Figuur 29 Tabblad in een DGWM MHWp5 project



Voor een Hydra MHWp5 project zijn dat de tabbladen ‘SGWM’ en ‘HYDRA’. De functionaliteit van het tabblad ‘SGWM’ is identiek aan SGWM voor een willekeurig rekenmodel en het tabblad ‘HYDRA’ is gelijk van opzet aan dat van ‘WGWM’ in SGWM voor een willekeurig rekenmodel. In het tabblad ‘Hydra’ worden invoerbestanden aangemaakt waarmee met een Hydra softwarepakket probabilistische berekeningen kan uitvoeren.

Figuur 30 Tabblad in een Hydra MHWp5 project



De basis functionaliteit van SGWM is verder overal van toepassing (zie ook hoofdstuk 3). Voor elk rekenmodel is er een apart MD bestand en ook de rekeninstellingen verschillen per rekenmodel.

5 Referenties

Kappe A. en Thonus B.I., 2017.
Functioneel ontwerp NWM - Waterveiligheid. Rijkswaterstaat WVL. Versie 1.0. Albert Kappe (SSC Campus) en Bart Thonus (HKV lijn in water). 14 augustus 2017.

Thonus B.I., 2019.
SGWM voor MHWp5 - Gebruikershandleiding. Rijkswaterstaat WVL. Bart Thonus (HKV lijn in water). Mei 2019. HKV project PR3789.20

Bijlagen

A SGWM-projectbestand

Het SGWM-projectbestand beschrijft de status van een SGWM-project en bevat verwijzingen naar alle bestanden die gebruikt worden in het betreffende project. Bij het openen van een bestaand SGWM-project moet een SGWM-projectbestand (*.SGWM) geselecteerd worden. Afhankelijk van de beschikbaarheid van informatie in dit bestand wordt het project.

Het SGWM-projectbestand is een XML-bestand en kan geopend en aangepast worden in een willekeurige tekst-editor of XML-editor. Geadviseerd wordt om minimaal gebruik te maken van een tekst-editor met XML-tag herkenning (bijvoorbeeld Notepad++), wat de kans op fouten bij het aanpassen van een XML-bestand verkleint.

Hieronder is een voorbeeld gegeven van een SGWM-projectbestand[33] zoals dat aangemaakt wordt door de applicatie SGWM bij het aanmaken van een nieuw project (zie ook hoofdstuk 3.2).

Projectbestand na aanmaken van een nieuw project

```
<?xml version="1.0" encoding="utf-8"?>
<SGWM>
	<Version>2.1.0</Version>
	<Name>Swan-EUR.sgwm</Name>
	<Folder>d:\users\thonus\mijnsoftware\sgwm_deltares\trunk\
		tests\user_examples\swan - europoort\
		swan-eur-hr2011\sgwm</ProjectDefinitionFile>
	<ProjectDefinitionFile>eur_swan_projectdefinitiebestand.xml
		</ProjectDefinitionFile>
	<WGWMModelDefinitionFile></WGWMModelDefinitionFile>
	<WGWMCalculationSettings></WGWMCalculationSettings>
	<CalculationSet></CalculationSet>
	<Calculations></Calculations>
```

Het versienummer van de gebruikte SGWM-software staat tussen de tags <Version>. De naam het van het SGWM-projectbestand staat tussen de tags <Name> en de folder waarin projectbestanden worden opgeslagen tussen de tags <Folder>. Het gekozen project-definitie-bestand (bestandsnaam inclusief volledig pad) staat tussen de tags <ProjectDefinitionFile> (zie ook paragraaf 2.2).

Let op dat elk item in een XML altijd begint met een tag tussen <> en afgesloten wordt met </>!

De overige tags in het projectbestand hebben betrekking op:

- <WGWMModelDefinitionFile>: het MDB met instellingen voor het rekenmodel (zie paragraaf 2.3).
- <CalculationSettings>: een verwijzing naar een bestand met de gekozen rekeninstellingen in SGWM (zie bijlage D).
- <CalculationSet>: een verwijzing naar het bestand met de definitie van een sommenset (zie bijlage B). Als dit bestand beschikbaar is en het project wordt geopend dan wordt het tabblad ‘sommen’ geactiveerd;

[33] Het bestand is van het type XML en gebruikt standaard UTF-8 encoding.

- <Calculations>: een verwijzing naar een CSV-bestand met de daadwerkelijk sommenset (zie bijlage C);

B Definitie van de sommenset: Sommenset.xml

In het sommenset-definitie-bestand wordt de definitie van de sommenset opgeslagen zoals deze in gebruikersschil van SGWM gedefinieerd kan worden (zie paragraaf 3.4). Als dit bestand bestaat wordt de inhoud van dit bestand gebruikt om de componenten van het scherm 'SGWM te laden.

Bij wijzigingen via de gebruikersschil wordt gevraagd om deze wel of niet op te slaan in het sommenset-definitie-bestand. Het bestand kan ook buiten het gebruik van de gebruikersschil van SGWM aangepast worden.
Let op: de keuzevelden bevatten keuzeopties op basis van beschikbaarheid van modelinvoer. Aanpassing buiten de gebruikersschil om vergt goed inzicht van de beschikbaarheid van de modelinvoer om fouten te voorkomen.

Het sommenset-definitie-bestand is een XML-bestand en kan geopend en aangepast worden in een willekeurige tekst-editor of XML-editor. Geadviseerd wordt om minimaal gebruik te maken van een tekst-editor met XML-tag-herkenning (bijvoorbeeld Notepad++), wat de kans op fouten bij het aanpassen van een XML-bestand verkleint.

Hieronder is een voorbeeld gegeven van een sommenset-definitie-bestand[34] (zie ook hoofdstuk 3.4).

Sommenset definitie

```
<?xml version="1.0" encoding="utf-8"?>
<CalculationSet>
	<Parameter>
		<Name>Windsnelheid (potentieel) in m/s</Name>
		<Key>U</Key>
		<Values>43</Values>
	</Parameter>
	<Parameter>
		<Name>Windrichting in graden tov Noord</Name>
		<Key>D</Key>
		<Values>045</Values>
		<Values>247</Values>
	</Parameter>
	<Parameter>
		<Name>Waterstand in cm tov NAP</Name>
		<Key>L</Key>
		<Values>m100</Values>
		<Values>p100</Values>
		<Values>p900</Values>
	</Parameter>
</CalculationSet>
```

Tussen de tags <CalculationSet> zijn de parameters uit het PDB opgenomen (met naam <Name> en SGWM Key <Key>) inclusief de waarde(n) die aan de definitie van de sommenset zijn toegekend. Dit is altijd een deelverzameling van de mogelijke waarden per parameter uit de parameterdefinitie uit het PDB.

[34] Het bestand is van het type XML en gebruikt standaard UTF-8 encoding.

C Overzicht sommen: sommen.csv

Na het definiëren van een sommenset (zie ook Bijlage B) worden voor alle mogelijke parametercombinaties sommen aangemaakt. In de gebruikersschil van SGWM worden alle sommen in een tabel gepresenteerd. De inhoud van deze tabel wordt ook weggeschreven naar het CSV-bestand 'sommen.csv'. Als dit bestand bestaat wordt de inhoud van dit bestand ook gebruikt om de tabel te laden in het scherm 'Sommen' van de gebruikersschil.

De inhoud van de tabel kan via de gebruikersschil van SGWM niet aangepast worden (behalve door een nieuwe definitie van de sommenset aan te maken in het scherm 'Sommenset'). Buiten de gebruikersschil van SGWM om is het wel mogelijk om aanpassingen door te voeren. Aan het eind van paragraaf 3.5 is beschreven dat het mogelijk is om handmatig sommen uit de sommenset te verwijderen buiten de gebruikersschil van SGWM om. Dit om bijvoorbeeld sommen met ongewenste parametercombinaties, zoals hoge zeewaterstanden bij wind uit het oosten, of een windsnelheid van 0 m/s voor alle windrichtingen, uit de sommenset te verwijderen.

Het bestand met de sommenset is een CSV-bestand en kan geopend en aangepast worden in een willekeurige tekst-editor of in een spreadsheet programma, zoals MS Excel.

Hieronder is een voorbeeld gegeven van een sommenset-bestand.

Sommenset

```
"Naam";"Q";"Status SGWM";"Status WGWM";"Datum WGWM";"Checked
WGWM"
"rijn_QS00600";"600";"Gereed";"invoer gereed";"19-12-2023
13:17:10";2
"rijn_QS02000";"2000";"Gereed";"invoer gereed";"19-12-2023
13:17:10";2
"rijn_QS04000";"4000";"Gereed";"invoer gereed";"19-12-2023
13:17:10";2
"rijn_QD06000";"6000";"Gereed";"invoer gereed";"19-12-2023
13:17:10";2
```

Het CSV-bestand bevat alle kolommen uit de tabel van het scherm 'Sommen' uit de gebruikersschil van SGWM, met uitzondering van het unieke volgnummer. De kolomtitels zijn in de eerste rij weergegeven, gescheiden door een puntkomma. Alle navolgende rijen komen overeen met één som. De laatste kolom komt overeen met de checkboxes in de tabel met sommen. In het CSV-bestand komt 0 overeen met niet aangevinkt en 2 met wel aangevinkt.

D Gekozen rekeninstellingen: Rekeninstellingen.xml

Initieel worden de rekeninstellingen geladen vanuit het MDB (zie ook paragraaf 2.3.7). De gebruiker kan deze instellingen aanpassen in het scherm met rekeninstellingen (zie ook hoofdstuk 3.5). De door de gebruiker gekozen rekeninstellingen worden opgeslagen in het bestand (WGWM_)Rekeninstellingen.xml.

Het bestand met rekeninstellingen is een XML-bestand en kan geopend en aangepast worden in een willekeurige tekst-editor of XML-editor. Geadviseerd wordt om minimaal gebruik te maken van een tekst-editor met XML-tag-herkenning (bijvoorbeeld Notepad++), wat de kans op fouten bij het aanpassen van een XML-bestand verkleint.

Hieronder is een voorbeeld gegeven van een bestand met rekeninstellingen.

Rekeninstellingen

```
<?xml version="1.0" encoding="utf-8"?>
<CalculationSettings>
	<CalculationSetting>
		<Name>Queue</Name>
		<Value>RWS_normal</Value>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Docker container</Name>
		<Value>deltares/swan:41.31A.1_1.0.0</Value>
	</CalculationSetting>
	<CalculationSetting>
		<Name>Docker parallel</Name>
		<Value>4</Value>
	</CalculationSetting>
</CalculationSettings>
```

Voor elke rekeninstelling is de Naam (aangeduid met de tags <Name> en gelijk aan de naam uit het MDB) en de, door de gebruiker gekozen, waarde (aangeduid met de tags <Value>) weggeschreven.

E Logbestand SGWM

Voor elk SGWM-project wordt een logbestand aangemaakt. Dit logbestand krijgt de naam van het SGWM-project met de extensie '.log'.

In het logbestand wordt informatie weggeschreven bij het genereren van invoer van sommen (zie paragraaf 3.5). Er worden verschillende categorieën logmeldingen weggeschreven:

- INFO een melding die een statusverandering beschrijft (bijvoorbeeld het aanmaken van een folder voor een som).
- ERROR fouten die geconstateerd worden door SGWM, bijvoorbeeld omdat in een template verwezen wordt naar een bestand dat niet beschikbaar is.
- DEBUG extra detail logmeldingen over status die weggeschreven worden als een debug-optie aangezet wordt. Dit zijn geen foutmeldingen!

Hieronder is een voorbeeld gegeven van een logbestand.

Log bestand

```
2018-03-02 15:24:15,060 - INFO - Create folder:
KsrMp040Q1850U10D225
2018-03-02 15:24:15,076 - ERROR - [Errno 2] No such file or
directory: 'P:\\ijssel-vechtdelta\\waqua-ijvd-hr2017_5-
v4\\hr2017_5-v4\\computations\\hr2017_wda\\templates\\options.sh'
2018-03-02 15:24:15,076 - ERROR - 2
2018-03-02 15:25:32,895 - INFO - Create folder:
KsrMp040Q1850U10D225
2018-03-02 15:25:32,926 - DEBUG - template : options.sh copied
2018-03-02 15:25:32,958 - DEBUG - template : getdata.sh copied
2018-03-02 15:25:32,958 - DEBUG - input template :
siminp.basis_ijvd-v4
```

Controleer elke keer na het genereren van invoer van sommen of er geen fouten zijn geconstateerd in het logbestand.

Let op: het logbestand wordt eenmalig aangemaakt bij het beginnen van een nieuw project. Daarna wordt het bestand uitgebreid met nieuwe logmeldingen elke keer als er invoer van sommen wordt gegenereerd. Wanneer het aantal sommen heel groot is kan het logbestand groot worden. Geadviseerd wordt om het logbestand af en toe te verwijderen.

F Folderstructuur modelschematisaties

F.1 Inleiding

Rijkswaterstaat en Deltares (beheerder van modelsoftware en model-schematisaties uit het NWM) hebben onderling afspraken gemaakt over de folderstructuur die toegepast wordt voor modelschematisaties. Deze bijlage geeft het actueel overzicht (op het moment van opstellen van deze gebruikershandleiding) van deze afspraken. Uitgangspunt is dat de afspraken generiek zijn en model-overstijgend.

F.2 Uitgangspunten

Bij het samenstellen van de folderstructuur voor modelschematisaties worden de volgende uitgangspunten gehanteerd:

1. Op het hoogste niveau worden de gebieden onderscheiden, bijvoorbeeld ‘ijssel-vechtdelta’, ‘maas’, ‘ijsselmeer’, etc.
2. Binnen elk gebied kunnen één of meerdere modelschematisaties onderscheiden worden. Een modelschematisatie wordt in eerste instantie gekenmerkt door een gebied, maar ook de toepassing (de modelsoftware) waarvoor de modelschematisatie is opgezet. De naam van de modelschematisatie wordt samengesteld uit vier onderdelen (gescheiden door ‘-’):
   e. De naam van de modelsoftware, bijv. ‘waqua’;
   f. De naam van het gebied, bijv. ‘ijvd’;
   g. De naam/omschrijving van de schematisatie, bijv. ‘hr2017_5’’;
   h. En een versienummer, bijv. ‘v4’.

   Tezamen geeft dit bijvoorbeeld een unieke naam voor de modelschematisatie, te weten ‘waqua-ijvd-hr2017_5-v4’.
3. Binnen een model worden vier onderdelen onderscheiden:
   a. Bestanden die het statische deel van de schematisatie beschrijven geclusterd in de map ‘geometry’. In deze map wordt in ieder geval onderscheid gemaakt in een map ‘bodem’ met het bodemgrid en een map ‘locaties’ met de definitie van locaties in het grid. Daarnaast kunnen er verschillende submappen opgenomen zijn met andere statische onderdelen van de schematisatie, zoals ‘schotjes’, ‘randen’, ‘kunstwerken’, etc.
      Een schematisatie kent altijd maar één variant van deze bestanden; m.a.w. er is maar 1 gebiedschematisatie. Als er een andere gebiedschematisatie bestaat dan wordt er één niveau hoger een nieuwe map aangemaakt (zie punt 2).
   b. Bestanden die de randvoorwaarden van het model beschrijven in de map ‘boundary_conditions’. Hierin kan onderscheid gemaakt worden in bijvoorbeeld bestanden voor de wind (bijv. in de submap ‘wind’) en voor afvoeren en waterstanden (bijv. in de submap ‘flow’).

Het is mogelijk dat er verschillende randvoorwaardensets beschikbaar zijn voor een modelschematisatie, bijv. 'hr2017' en 'hr2020'. Deze zijn beide in combinatie met de gebiedschematisatie uit punt 3a toe te passen.

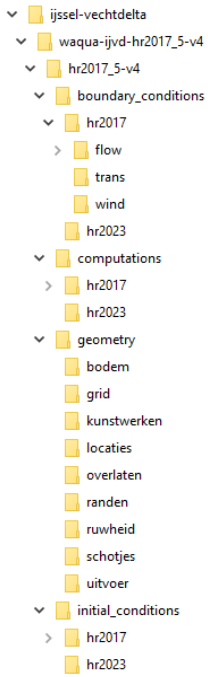
c. Bestanden die de initiële conditie(s) van een model beschrijven (folder initial_conditions). Dit zijn in de regel resultaten van een eerder uitgevoerde som met als doel de initial state van het model vast te leggen. Elke combinatie van een gebiedschematisatie met een randvoorwaardenset verwijst naar één of meerdere initiële condities. Het is dus niet zomaar mogelijk om elke initiële conditie op elke combinatie van randvoorwaarden en gebiedschematisatie toe te passen.

d. De folder 'computations' bevat een map met templates die in die door de SGWM module gebruikt worden om per som (telkens een aparte map in de folder 'computations', gekenmerkt door een unieke ID bestaande uit de paramatercode en parameterwaarde voor alle parameters) klaar te zetten ten behoeve van de modelsom met behulp van een scheduler (bijv. LSF).

Zoals hierboven beschreven zijn er afhankelijkheden tussen 'boundary_conditions', 'intial_conditions' en 'computations'. Deze relaties worden afgedwongen in de naamgeving van de submappen. Bijvoorbeeld de submap 'hr2017' in 'boundary_conditions' bevat bestanden die horen bij de submap 'hr2017' in 'intial_conditions' en de rekenresultaten komen in de submap 'hr2017' in 'computations'.

F.3 Voorbeeld

Onderstaande figuur toont een voorbeeld van een folderstructuur voor een modelschematisatie van de IJssel-Vechtdelta.



Figuur 31 Voorbeeld folderstructuur modelschematisaties

G XSD-definitie van een PDB en MDB

Deze bijlage bevat de XSD-definitie van respectievelijk het PDB en MDB (voor SGWM versie 2.1.0)

Onderstaande tabel bevat de XSD-definitie van het PDB.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="ParameterType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="float" />
      <xs:enumeration value="string" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="PDB">
    <xs:annotation>
      <xs:documentation>PDB stands for Project-Definitie-
Bestand</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Name" type="xs:string">
          <xs:annotation>
            <xs:documentation>a random name for the PDB (is only
metadata)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="Instrumentarium" type="xs:string">
          <xs:annotation>
            <xs:documentation>the name of the calculation environment used in
this PDB (is only metadata)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="ModelSoftwarePackage">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Name" type="xs:string">
                <xs:annotation>
                  <xs:documentation>name of the model sofware, e.g.: Swan,
Waqua, Sobek, D-Hydro, Hydra-NL, etc. (is only
metadata)</xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element name="Version" type="xs:string">
                <xs:annotation>
                  <xs:documentation>version number of the model software (is
only metadata)</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Modelschematisation">
          <xs:annotation>
            <xs:documentation>definition of the model
schematisation</xs:documentation>
          </xs:annotation>
          <xs:complexType>
```

```
        <xs:sequence>
          <xs:element name="Name" type="xs:string">
            <xs:annotation>
              <xs:documentation>unique name (including version number) of
the model schematisation (is only metadata)</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="Folder" type="xs:string">
            <xs:annotation>
              <xs:documentation>root path to all files of the model
schematisation (must be an existing path on the
system)</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="Geometry" minOccurs="0">
            <xs:annotation>
              <xs:documentation>optional</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Folder" type="xs:string">
                  <xs:annotation>
                    <xs:documentation>folder name with geometric model files;
relative to the root path</xs:documentation>
                  </xs:annotation>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="BoundaryCondition" minOccurs="0">
            <xs:annotation>
              <xs:documentation>optional</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element type="xs:string" name="Folder">
                  <xs:annotation>
                    <xs:documentation>folder name with boundary model
conditions; relative to the root path</xs:documentation>
                  </xs:annotation>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="InitialCondition" minOccurs="0">
            <xs:annotation>
              <xs:documentation>optional</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element type="xs:string" name="Folder">
                  <xs:annotation>
                    <xs:documentation>folder name with initial model
conditions; relative to the root path</xs:documentation>
                  </xs:annotation>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Computation">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Folder" type="xs:string">
```

```
                <xs:annotation>
                  <xs:documentation>folder name with result input files of
SGWM; relative to the root path</xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element name="Id" type="xs:string">
                <xs:annotation>
                  <xs:documentation>compuatation id with result input files
of SGWM; relative to the root path</xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element name="Parameters">
                <xs:annotation>
                  <xs:documentation>definition of parameters which vary per
computation (the unique combination of all parameter values defines the
complete calculation set)</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Parameter" maxOccurs="unbounded"
minOccurs="1">
                      <xs:annotation>
                        <xs:documentation>parameter definition  parameter
definition</xs:documentation>
                      </xs:annotation>
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element type="xs:string" name="Name">
                            <xs:annotation>
                              <xs:documentation>a random name/description for
a parameter</xs:documentation>
                            </xs:annotation>
                          </xs:element>
                          <xs:element type="xs:string" name="Key">
                            <xs:annotation>
                              <xs:documentation>unique parameter key to be
used in different template files (as SGWM key)</xs:documentation>
                            </xs:annotation>
                          </xs:element>
                          <xs:element type="ParameterType" name="Type">
                            <xs:annotation>
                              <xs:documentation>type of parameter values (float
or string)</xs:documentation>
                            </xs:annotation>
                          </xs:element>
                          <xs:element name="Values">
                            <xs:annotation>
                              <xs:documentation>array of values for the
parameter </xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element type="xs:string" name="Value"
maxOccurs="unbounded" minOccurs="1"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
```

```
                <xs:element name="Variables" minOccurs="0">
                  <xs:annotation>
                    <xs:documentation>from parameters derived
values</xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="Variable" maxOccurs="unbounded"
minOccurs="1">
                        <xs:annotation>
                          <xs:documentation>variable
definition</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element type="xs:string" name="Name">
                              <xs:annotation>
                                <xs:documentation>a random name/description for
a variable</xs:documentation>
                              </xs:annotation>
                            </xs:element>
                            <xs:element type="xs:string" name="Key">
                              <xs:annotation>
                                <xs:documentation>unique variable key to be used
in different template files (as SGWM key)</xs:documentation>
                              </xs:annotation>
                            </xs:element>
                            <xs:element type="ParameterType" name="Type">
                              <xs:annotation>
                                <xs:documentation>type of parameter values (float
or string)</xs:documentation>
                              </xs:annotation>
                            </xs:element>
                            <xs:element name="Values">
                              <xs:annotation>
                                <xs:documentation>array with values of variable
depending on specific values of one or more
parameters</xs:documentation>
                              </xs:annotation>
                              <xs:complexType>
                                <xs:sequence>
                                  <xs:element type="xs:string" name="Value"
maxOccurs="unbounded" minOccurs="1"/>
                                </xs:sequence>
                              </xs:complexType>
                            </xs:element>
                          </xs:sequence>
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Onderstaande tabel bevat de XSD-definitie van het MDB.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="OperatorType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="LT" />
      <xs:enumeration value="LE" />
      <xs:enumeration value="EQ" />
      <xs:enumeration value="GE" />
      <xs:enumeration value="GT" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="TemplateType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="inputfile" />
      <xs:enumeration value="lsf" />
      <xs:enumeration value="bsub" />
      <xs:enumeration value="general" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="CalculationSettingType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="float" />
      <xs:enumeration value="string" />
      <xs:enumeration value="integer" />
      <xs:enumeration value="combo" />
      <xs:enumeration value="file" />
      <xs:enumeration value="folder" />
      <xs:enumeration value="datetime" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="MDB">
    <xs:annotation>
      <xs:documentation>MDB stands for Model-Definitie-
Bestand</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Name" type="xs:string" />
        <xs:element name="Computation">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Templates">
                <xs:annotation>
                  <xs:documentation>add minimal one inputfile and a bsub and
lsf type template</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="Folder">
                      <xs:annotation>
                        <xs:documentation>folder name with templates; relative to
the root path</xs:documentation>
                      </xs:annotation>
                    </xs:element>
                    <xs:element name="Template" maxOccurs="unbounded"
minOccurs="3">
                      <xs:annotation>
```

```
                    <xs:documentation>definition of a template
file</xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element type="TemplateType" name="Type">
                        <xs:annotation>
                          <xs:documentation>template type: inputfile (template
is model inputfile), lsf (template is lsf job script), bsub (template is bsub
commando script) or general (all other templates)</xs:documentation>
                        </xs:annotation>
                      </xs:element>
                      <xs:element name="File">
                        <xs:annotation>
                          <xs:documentation>name of template file in template
folder</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                          <xs:simpleContent>
                            <xs:extension base="xs:string">
                              <xs:attribute name="shell" type="xs:string"
use="optional">
                              </xs:attribute>
                            </xs:extension>
                          </xs:simpleContent>
                        </xs:complexType>
                      </xs:element>
                      <xs:element name="Conditions" minOccurs="0">
                        <xs:annotation>
                          <xs:documentation>table with specific conditions when
to copy the template file to a computation folder</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                          <xs:sequence>
                            <xs:element name="Condition"
maxOccurs="unbounded" minOccurs="0">
                              <xs:annotation>
                                <xs:documentation>with attribute
calculationsetting="true" indicating to use a calculationsetting as key for
condition; false (or attribute ommitted) indicates a parameter or variable
key</xs:documentation>
                              </xs:annotation>
                              <xs:complexType>
                                <xs:sequence>
                                  <xs:element type="xs:string" name="Key">
                                    <xs:annotation>
                                      <xs:documentation>parameter, variable or
calculationsetting key for condition </xs:documentation>
                                    </xs:annotation>
                                  </xs:element>
                                  <xs:element type="xs:string" name="Value">
                                    <xs:annotation>
                                      <xs:documentation>parameter, variable or
calculationsetting value for condition</xs:documentation>
                                    </xs:annotation>
                                  </xs:element>
                                  <xs:element type="OperatorType"
name="Operator">
                                    <xs:annotation>
                                      <xs:documentation>operator e.g. EQ (value
must be equal), LT (less than), GT (greater than), LE (less or equal) and GE
(greater of equal)</xs:documentation>
                                    </xs:annotation>
```

```
                                </xs:element>
                               </xs:sequence>
                               <xs:attribute type="xs:string"
name="calculationsetting" use="optional"/>
                              </xs:complexType>
                             </xs:element>
                            </xs:sequence>
                           </xs:complexType>
                          </xs:element>
                          <xs:element type="xs:string" name="Update">
                           <xs:annotation>
                            <xs:documentation>true: update content of file with
SGWM keys (parameters, variables or calculationsettings); false: no update
of content, just copy the template file to the computation
folder</xs:documentation>
                           </xs:annotation>
                          </xs:element>
                         </xs:sequence>
                        </xs:complexType>
                       </xs:element>
                      </xs:sequence>
                     </xs:complexType>
                    </xs:element>
                    <xs:element name="Input">
                     <xs:complexType>
                      <xs:sequence>
                       <xs:element name="Folder" type="xs:string" />
                       <xs:element name="File" type="xs:string" minOccurs="1"/>
                      </xs:sequence>
                     </xs:complexType>
                    </xs:element>
                    <xs:element name="States">
                     <xs:complexType>
                      <xs:sequence>
                       <xs:element name="State" maxOccurs="unbounded"
minOccurs="1">
                        <xs:complexType>
                         <xs:sequence>
                          <xs:element name="Name" type="xs:string" />
                          <xs:element name="Text" type="xs:string" />
                          <xs:element name="File" type="xs:string" />
                          <xs:element name="Folder" type="xs:string"/>
                          <xs:element name="Progress" type="xs:boolean" />
                         </xs:sequence>
                        </xs:complexType>
                       </xs:element>
                      </xs:sequence>
                     </xs:complexType>
                    </xs:element>
                   </xs:sequence>
                  </xs:complexType>
                 </xs:element>
                 <xs:element name="CalculationSettings">
                  <xs:complexType>
                   <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="CalculationSetting"
minOccurs="0">
                     <xs:complexType>
                      <xs:sequence>
                       <xs:element name="Name" type="xs:string" />
                       <xs:element name="Key" type="xs:string" />
                       <xs:element name="Type" type="CalculationSettingType" />
                       <xs:element name="Conditions" minOccurs="0">
```

```
                <xs:annotation>
                  <xs:documentation>table with specific conditions when to
copy the template file to a computation folder</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="Condition"
minOccurs="0">
                      <xs:annotation>
                        <xs:documentation>with attribute
calculationsetting="true" indicating to use a calculationsetting as key for
condition; false (or attribute ommitted) indicates a parameter or variable
key</xs:documentation>
                      </xs:annotation>
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="CalculationSettings">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="CalculationSetting">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="Key" type="xs:string">
                                        <xs:annotation>
                                          <xs:documentation>parameter, variable or
calculationsetting key for condition </xs:documentation>
                                        </xs:annotation>
                                      </xs:element>
                                      <xs:element name="Value" type="xs:string">
                                        <xs:annotation>
                                          <xs:documentation>parameter, variable or
calculationsetting value for condition</xs:documentation>
                                        </xs:annotation>
                                      </xs:element>
                                      <xs:element name="Operator"
type="xs:string">
                                        <xs:annotation>
                                          <xs:documentation>operator e.g. EQ
(value must be equal), LT (less than), GT (greater than), LE (less or equal)
and GE (greater of equal)</xs:documentation>
                                        </xs:annotation>
                                      </xs:element>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                          <xs:element name="Value" type="xs:string"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element minOccurs="0" name="Items" type="xs:string"
/>
              <xs:element minOccurs="0" name="Value" type="xs:string"
/>
              <xs:element minOccurs="0" name="Edit" type="xs:boolean"
/>
            </xs:sequence>
          </xs:complexType>
```

```
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

H Putty-commando’s

Hieronder is een (beperkt) aantal veel gebruikte Putty commando’s weergegeven in combinatie met LSF (zoals gebruikt kan worden in het ModellenPlatform).

ls	geeft een overzicht van alle folders in de actieve directory
cd *naam*	navigeert naar een specifieke subfolder
cd ..	navigeert één folder omhoog
cd -	navigeert één folder terug
bhost	geeft een overzicht van alle nodes in het rekengrid inclusief hun status
bjobs	geeft een overzicht van alle jobs inclusief hun status
bjobs -r	geeft een overzicht van alle jobs die gestart zijn inclusief hun status
watch bjobs	een overzicht met jobs wordt continu ververst (met CTRL-C kan dit overzicht weer geannuleerd worden)
./*naam*	start het bsub batch bestand (*naam* is gelijk aan de naam van dit bestand) en submit alle jobs naar LSF
history	overzicht met recente commando’s (selecteren met de muis, plakken met rechter muisknop)
lshosts	per beschikbare node wordt type OS gegeven (NTX64=Windows, X86_64=Linux)

Hoofdkantoor
HKV lijn in water BV
Botter 11-29
8232 JN Lelystad
Postbus 2120
8203 AC Lelystad

Nevenvestiging
Elektronicaweg 12
2628 XG Delft

0320 29 42 42
info@hkv.nl
www.hkv.nl