Plan van Aanpak voor Domein Decompositie bij WL en RIKZ

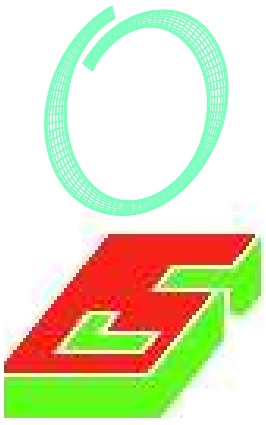
Technical Report TR97-05

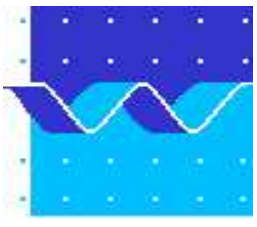
Date

28 november 1997

Author(s)

ir. S. Hummel, WL Delft,
dr.ir. M.R.T. Roest, VORtech Computing/SEPRA BV,
dr.ir. E.A.H. Vollebregt, VORtech Computing/SEPRA BV.





VORtech Computing
INGENIEURSBUREAU SEPRA
waterloopkundig laboratorium WL

Samenvatting

Zowel bij RIKZ als bij WL bestaat behoefte aan *proces-model koppeling* (bijv. stroming-morfologie), *domein decompositie* (bijv. het gebruik van verschillende roosterafstanden in verschillende delen van een model) en *parallel rekenen* (het inzetten van meerdere computers om een berekening sneller gedaan te krijgen). Deze drie behoeftes zijn verschillende vormen van de wens om afzonderlijke rekenprogramma's te kunnen koppelen.

Om parallel rekenen mogelijk te maken is in het verleden in opdracht van RIKZ door de TU Delft het CouPle concept ontwikkeld en geïmplementeerd. Op basis van dit concept kunnen ook domein decompositie en proces-model koppeling betrekkelijk eenvoudig gerealiseerd worden. In het kader van het DDPR01 project is de implementatie van CouPle uitgebouwd ten behoeve van domein decompositie. Daarmee is aangetoond dat het concept zich inderdaad leent andere toepassingen dan alleen parallel rekenen.

Bij WL is het Delft-Hydra concept ontwikkeld en geïmplementeerd, vooral met het oog op domein decompositie en proces-model koppeling. De ervaringen die tot nu toe met dit systeem opgedaan zijn, geven aan dat het concept inderdaad de gewenste flexibiliteit biedt.

In dit rapport wordt voor beide instituten geschetst hoe de verdere ontwikkeling van respectievelijk de CouPle en Hydra concepten kan verlopen. Er worden concrete actiepunten aangegeven op basis waarvan op korte termijn kan worden toegewerkt naar een operationalisatie van in ieder geval domein decompositie.

Het zal duidelijk zijn dat de ontwikkelingen bij RIKZ en bij WL sterk gerelateerd zijn. Dit biedt de mogelijkheid om van elkaars kennis en ervaring gebruik te maken. Bovendien is in het DDPR01 project een feitelijke koppeling tot stand gebracht tussen de CouPle en Hydra omgevingen. Daarmee is het in principe mogelijk geworden om WL-programmatuur onder Delft-Hydra aan RIKZ programmatuur op basis van CouPle te koppelen. In dit rapport wordt concreet aangegeven welke mogelijkheden dit biedt voor verdere samenwerking tussen WL en RIKZ.

Inhoudsopgave

Hoofdstuk 1

Introductie

Zowel bij het Waterloopkundig Laboratorium (WL) als bij het Rijksinstituut voor Kust en Zee (RIKZ) van Rijkswaterstaat bestaat er behoefte aan een aantal samenhangende ontwikkelingen ten aanzien van simulatie-modellen:

- *proces-model koppeling*: het gekoppeld kunnen simuleren van samenhangende fysische processen;
- *domein decompositie*: een ruimtelijke onderverdeling kunnen maken van het modeldomein en uitvoeren van gekoppelde simulaties op de deeldomeinen;
- *parallel rekenen*: het kunnen bekorten van de simulatieduur, door het inzetten van meerdere computers/rekeneenheden voor een taak.

Alle drie de behoeftes blijken in praktijk neer te komen op het kunnen koppelen van afzonderlijke rekenprogramma's. Door hiervoor faciliteiten te ontwikkelen kan relatief snel in de behoeftes worden voorzien en kan de bestaande simulatie-programmatuur grotendeels worden hergebruikt.

Beide instituten hebben de afgelopen jaren afzonderlijk een aantal inspanningen gepleegd en hebben een eigen aanpak ontwikkeld. Het WL heeft het Delft-Hydra concept ontwikkeld, en heeft hiermee domein decompositie met verticale verfijning gerealiseerd binnen TRISULA. RIKZ heeft de beschikking over een geparallelliseerde versie van TRIWAQ die is ontwikkeld in samenwerking met de TU Delft. Het hierbij gehanteerde concept heet CouPle, wat staat voor "coupled processes environment".

Zowel bij RIKZ als bij WL zijn de respectievelijke ontwikkelingen in volle gang. Omdat een overgang naar één gezamenlijk systeem niet op korte termijn haalbaar is, zullen beide ontwikkellijnen vooralsnog naast elkaar lopen. Dit project heeft echter duidelijk gemaakt dat

- het op dit moment technisch goed mogelijk is om de afzonderlijke systemen te koppelen, en
- er een zinvolle invulling gegeven kan worden aan verdere samenwerking op het gebied van domein decompositie.

In dit plan van aanpak zal geschetst worden hoe de ontwikkeling binnen RIKZ en WL vanaf dit punt kan vervolgen, niet alleen voor elk instituut apart maar vooral ook in onderlinge samenhang. Allereerst zal in Hoofdstuk 2 de ontwikkeling van domein decompositie binnen RIKZ nader worden uitgewerkt, gevolgd door eenzelfde uiteenzetting voor WL in Hoofdstuk 3. Vervolgens zal in Hoofdstuk 4 ingegaan worden op de samenhang tussen de twee ontwikkelingen.

Hoofdstuk 2

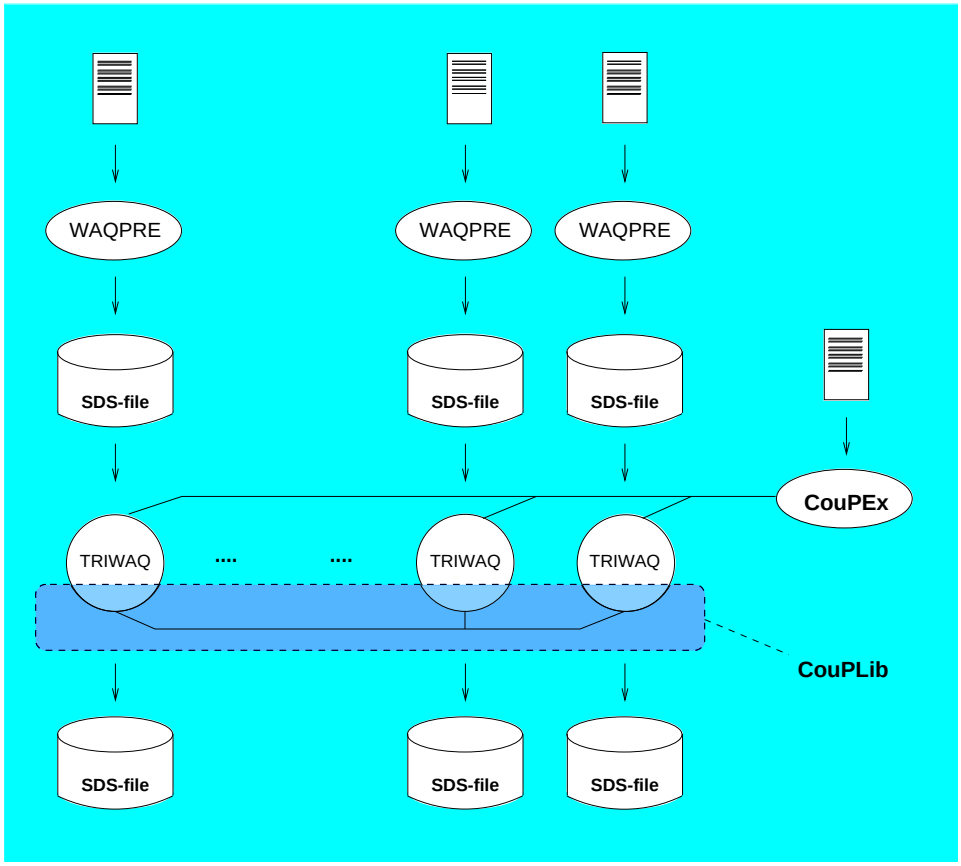
Aanpak van Domein Decompositie binnen RIKZ

2.1 Introductie van het CouPle concept

Figuur 2.1 geeft een overzicht van een gekoppelde run in het CouPle-concept. Enkele karakteristieken van de onderdelen van het concept zijn als volgt:

- Afzonderlijke SDS-files worden gecreëerd voor ieder rekenprogramma door de eigen preprocessors (of vanuit een SDS-file voor een globaal domein door het programma PARPRE).
- Een gekoppelde run begint met opstarten van CouPEx. Deze leest de beschrijving van de configuratie uit een tekstuele invoer-file (momenteel een SDS-file die is aangemaakt met het hulpprogramma CouPPre).
- CouPEx start de rekenprocessen een voor een op en stuurt hen dan de benodigde informatie over de andere rekenprocessen in het systeem. Gedurende de run vangt CouPEx de uitvoer naar het scherm op en verzorgt het fout-afhandeling. CouPEx heeft geen kennis van wat de rekenprocessen doen.
- De rekenprocessen communiceren met elkaar door middel van de communicatie-routines uit CouPLib. Hiervoor staan op de gewenste posities in de code eenvoudige aanroepen van CouPLib, zodanig dat zichtbaar is *wat* uitgewisseld wordt, maar niet *hoe*.
- De routines van CouPLib bevatten zo min mogelijk numerieke kennis. Ze bestaan uit aparte modules voor verschillende functies: interpoleren, converteren, communiceren. De routines worden gestuurd door "tabellen" die worden opgesteld met behulp van de benodigde arrays (laagdiktes) in de initialisatie. Hiervoor moet eventueel tijdens de initialisatie al enige informatie worden uitgewisseld.
- Het uitvoeren van convergentie-checks is een taak van de rekenprocessen. Hiervoor worden globale communicatie routines aangeboden door CouPLib.

Het CouPle concept heeft inmiddels zijn waarde bewezen voor parallel rekenen (zie bijvoorbeeld het rapport "Installation of Parallel TRIWAQ on a Cray T3E and an IBM SP2" [**?**]) en voor proces-model koppeling (zie het rapport "Tweeweg koppeling TRIWAQ-SIMPAR" [2]).



Figuur 2.1: *Schematisch overzicht van het* CouPle*-concept: afzonderlijke rekenprocessen die communiceren door het aanroepen van communicatie-routines en die worden opgestart door de koppelings-Executive.*

2.2 Aanpak voor verdere ontwikkeling

In dit project is aangetoond dat het CouPle concept inderdaad geschikt is voor het implementeren van domein decompositie voor de programmatuur van RIKZ. Bovendien is een belangrijk deel van de benodigde software-uitbreidingen geïmplementeerd in de vorm van een prototype. De ervaringen die daarmee opgedaan zijn, maken het mogelijk om aan te geven hoe vanuit de huidige stand van zaken op korte termijn bij RIKZ naar een operationeel bruikbare vorm van domein decompositie kan worden toegewerkt. Uitgangspunt hierbij vormt het rapport "Invoering Domein Decompositie RIKZ" [1], hierna aangeduid als "DD-rapport". In het overzicht hieronder zal steeds gerefereerd worden aan de gebruikerswensen zoals die in dat rapport zijn neergelegd.

Met de huidige structuur wordt voldaan aan gebruikerswensen Pre2 (gedeeltelijk), Pre4, Pre5, Pre6, Pre7 (gedeeltelijk), Sim1,Sim3,Sim7,Sim8 en Post2 uit het DD-rapport. In het volgende overzicht wordt aangegeven wat er nodig is om de huidige structuur te consolideren en aan meer gebruikerswensen tegemoet te komen. Het overzicht kan beschouwd worden als een nadere uitwerking van Fase 1 zoals omschreven in het DD-rapport.

- **Uitbreiden preprocessor voor DD randen**
  Een belangrijke beperking van de huidige versie van CouPled-TRIWAQ is dat de subdomeinen nog op een tamelijk omslachtige wijze uit een globaal rooster gegenereerd moeten worden. Voor het onmiddelijke vervolg van dit project zal het wenselijk zijn om de subdomeinen als aparte modellen aan te kunnen maken en te koppelen. Omdat bij koppelingsranden de lgrid tabel met een extra (virtueel) buitenpunt uitgebreid moet worden, moet hiervoor de preprocessor WAQPRE aangepast worden. Daarbij zal WAQPRE ook het concept van een koppelbare opening moeten gaan kennen (DD-rapport [1]: Pre7).
- **Uitbreidingen** CouPEx
  Op dit moment moet de configuratie van de gekoppelde run (welke rekenprocessen er zijn en bijvoorbeeld welke punten in de verschillende roosters gekoppeld zijn) in een SDS-file worden aangeleverd aan CouPEx. Het is wenselijk om het hulpprogramma CouPPre in CouPEx op te nemen zodat de tekstuele invoer direct wordt verwerkt. Bovendien zou CouPEx kunnen controleren dat de SDS-files van de subdomeinen consistent zijn met elkaar en met de opgegeven configuratie.
- **Uitbreiding koppelingsmogelijkheden**
  - **m:n koppeling horizontaal**
    Op dit moment is uitsluitend een 1:n koppeling in de verticale richting gerealiseerd. De daarbij ontwikkelde concepten zijn ook toepasbaar voor horizontale m:n koppelingen, maar specificatie en implementatie van deze koppelingen moet nog gebeuren.
  - **Verschillende koppelingsvergelijkingen** In dit project is een Dirichlet-Dirichlet koppeling gerealiseerd. Het zal echter, o.a. om redenen van convergentie snelheid, nodig zijn om ook over andere vormen van koppeling de beschikking te hebben (bijv. Neumann-Dirichlet).
- **Postprocessing**
  In het prototype is het niet mogelijk om de berekende resultaten in één SDS-file samen te voegen. De samenvoegings-tool (PARPOS) die voor parallel TRIWAQ ontwikkeld is, gaat er van uit dat alle subdomeinen hetzelfde aantal lagen hebben, wat bij domein decompositie zeker niet noodzakelijk het geval is. Het is de vraag of de structuur van de TRIWAQ LDS wel in staat is om meerdere submeshes met verschillende karakteristieken te herbergen. Wel is het mogelijk om de verschillende submeshes als aparte experimenten in een SDS-file onder te brengen, zodat er slechts één file meegenomen hoeft te worden naar de postprocessing. Vervolgens zullen er wel voorzieningen moeten worden geïmplementeerd voor het visualiseren van gekoppeld berekende resultaten. Principes zoals die voor PARPOS zijn ontwikkeld, kunnen hierbij zeer bruikbaar zijn. (DD-rapport [1]: Post1,Post3-5, voorbereiding op Pre9).

Als deze punten zijn uitgevoerd, zal het mogelijk zijn om los ontwikkelde modellen met verschillende (horizontale en verticale) roosterafstanden aan elkaar te koppelen op een numeriek efficiente en voor de gebruiker heldere manier. Bovendien zal een beperkte mogelijkheid bestaan voor globale visualisatie. Een en ander kan binnen een tijdsbestek van een half jaar gerealiseerd worden. Daarna is het waarschijnlijk wenselijk om eerst een evaluatie van de bestaande structuur te laten uitvoeren door eindgebruikers, zodat een duidelijk beeld ontstaat van de resterende knelpunten.

Voorts kan overwogen worden om te komen tot een consolidatie van de resultaten van het Particle project, waarin software is ontwikkeld ten behoeve van proces-model koppeling [2]. Deze software is op verschillende punten nog enigszins ad-hoc opgezet omdat ze is ontstaan in het kader van een onderzoeksproject.

Het is te verwachten dat verdere ontwikkelingen gewenst zullen zijn op de volgende punten:

- **Preprocessing**
  Het zal nodig worden om voorzieningen te implementeren waarmee gemakkelijk subdomeinen in een globaal domein gedefinieerd worden en van specifieke karakteristieken te voorzien. Dit is een slag die het beste nog vóór het aanroepen van WAQPRE kan worden ondergebracht, omdat WAQPRE ervan uitgaat dat alle karakteristieken (zoals aantal lagen, roosterafstand) bekend zijn. Het decomponeren van een model levert dan een set van invoer-files, die ieder apart met een normale WAQPRE preprocessing slag kunnen worden omgezet naar een SDS-file. (DD-rapport [1]: Pre1,Pre2,Pre3,Pre8).
- **Interactie met de Gebruiker**
  De gebruiker heeft op dit moment nog zeer beperkte mogelijkheden om de voortgang van de gekoppelde simulatie te monitoren en geen enkele mogelijkheid om het proces op een nette manier af te breken als het niet naar wens verloopt. Het ligt voor de hand om een apart gebruikers-interface proces te ontwikkelen, waarmee de gebruiker kan monitoren en sturen. Bovendien kan extra functionaliteit in de communicatie-bibliotheek worden ondergebracht om een nettere shut-down mogelijk te maken. (DD-rapport [1]: Sim1,Sim2,Sim4)
- **1:n tijdstapkoppelingen**
  Bij het werken met verschillende roosterafstanden is het wenselijk om ook per subdomein een andere tijdstap te kunnen hanteren, zodat niet het subdomein met het fijnste rooster zijn tijdstap opdrukt aan alle andere subdomeinen. Dit vereist onderzoek naar geschikte koppelings-vergelijkingen en bijbehorende koppelings-algoritmen. (DD-rapport [1]: Sim5)

Hoofdstuk 3

Aanpak van Domein Decompositie bij L

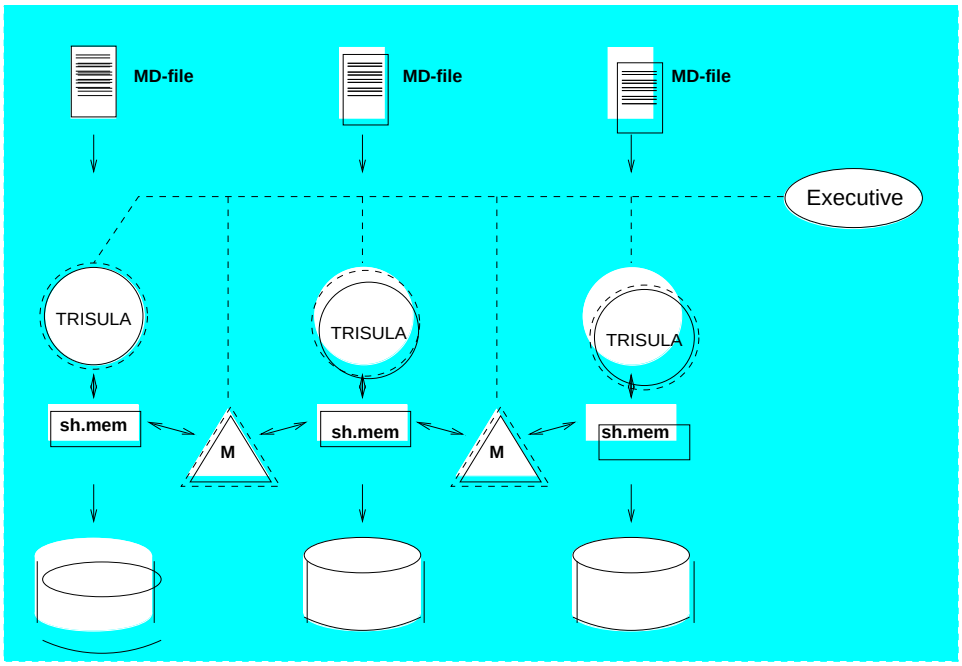
3.1 Introductie van het Delft-Hydra concept

Figuur 3.1 geeft een overzicht van domein decompositie binnen Delft-Hydra. De belangrijkste aspekten van de Delft-Hydra omgeving zijn:

- Voor elk subdomein wordt een md-file, de model definitie file, aangemaakt (dit gebeurt momenteel met de hand).
- De Hydra-Executive start eerst de Trisula processen (voor elk subdomein een proces), wacht tot deze allemaal door hun initialisatie-fase heen zijn, en start dan de Mapper processen.
- De communicatie tussen twee Trisula processen wordt verzorgd door de Mapper. Deze Mapper kan de in Shared Memory aanwezige data van zijn beide buurprocessen (de zgn. contexten) rechtstreeks benaderen
- De Mapper verzorgt het bouwen van DD-randvoorwaarden-vergelijkingen op de interface (in beide subdomeinen), het controleren van de convergentie van de oplossing, en - wanneer nog geen convergentie is bereikt - het opnieuwe opleggen van het rechterlid van de DD-randvoorwaarden-vergelijkingen.
- De Trisula-processen hebben er zelf nauwelijks weet van dat ze als subdomein gekoppeld zijn. Het gekoppeld zijn komt in de Trisula code tot uitdrukking in het aanroepen van de "NextStep"-call na diverse Build- en Solve- slagen.
- De Hydra-executive is - na het opstarten - in feite gedistribueerd aanwezig. Voor elk proces (Trisula of mapper) geldt dat ze met een volgende stap verder kunnen gaan zodra zijn buren hun voorgaande stap afgerond hebben. Alleen bij het controleren van de convergentie vindt *overall* synchronisatie plaats (alle mappers moeten convergentie hebben bereikt voordat naar een volgende stap kan worden gegaan).

3.2 Verdere ontwikkelingen

De Delft-Hydra omgeving (thans bestaande uit de executive en DD-Trisula) zal in de naaste toekomst worden uitgebreid op een aantal punten:



Figuur 3.1: *Schematisch overzicht van het Delft-Hydra-concept: afzonderlijke rekenprocessen, mappers (M) die de data-uitwisseling verzorgen en een executive die het geheel opstart en coördineert.*

- **Domein decompositie voor Delwaq.**
  Hiervoor wordt een DD-mapper gebouwd die vergelijkbaar is met die voor DD-Trisula.
- **Proces Decompositie (PD) voor Trisula en Delwaq.**
  Voor de overdracht van data tussen Trisula en Delwaq (PD) wordt een Proces Mapper gerealiseerd, en wordt de Hydra-executive uitgebreid met "time-step management".
- **Verfijning in de horizontaal**
  DD-Trisula zal worden voorzien van de mogelijkheid om subdomeinen met een verschillende roosterfijnheid te koppelen (waarbij per interface tussen twee subdomeinen geldt dat de verfijning m:n is, met m deelbaar door n, en dat 1 op de m/n roosterlijnen een doorgaande lijn is).

Naast genoemde acties zal het accent komen te liggen op de pre-processing en (in mindere mate) de post-processing rond Delft-Hydra.

- **User Interface**
  Op termijn zal de Delft-Hydra omgeving worden voorzien van een user interface (waarbinnen de gebruiker zijn processen op de diverse subdomeinen kan definieren). Dit user interface zal moeten worden geïntegreerd in dat van Delft3D.
- **Roostergeneratie**
  De roostergenerator die momenteel binnen WL wordt gebruikt (RgfGrid) biedt reeds de mogelijkheid om een gedeelte uit een bestaand rooster te knippen. Daarbij moeten overigens een aantal aantal acties verricht worden (m.n. het zorgen dat er een overlappende rij of kolom van virtuele punten aanwezig is), die beter ondersteund moeten worden. Hiertoe

zal RgfGrid worden uitgebreid met de mogelijkheid om in een bestaand rooster subdomeinen aan te wijzen, waarna automatisch de verschillende subroosters worden gegenereerd, alsmede de administratie voor de verschillende DD-mappers die tussen de subdomeinen aanwezig zijn, en informatie over de opsplitsing van de MD-files (zie volgend punt).

- **Invoer voor Trisula (MD-file)** De invoer voor Trisula wordt beschreven door middel van een MD-file. Bij opsplitsing van een domein in subdomeinen zal per subdomein een MD-file moeten worden aangemaakt, waarbij een (groot) aantal gegevens van elk subdomein zal afwijken van die in het oorspronkelijke domein (waarbij het overigens voornamelijk gaat om de M,N-indices in de subroosters van de diverse bijzondere cellen, zoals randvoorwaarden, dry-points, etc.). Om deze "vertaalslag" van MD-files voor de gebruiker hanteerbaar te maken zal er een ondersteunend tool moeten komen dat op basis van de oorspronkelijke MD-file en de door de roostergenerator gegenereerde "opknip"-informatie de de juiste subdomein-MD-files aanmaakt.
- **Postprocessing**
  Het binnen WL gehanteerde post-processings-tool GPP is reeds in staat om de 2D-waarden van meerdere subdomeinen binnen één plot weer te geven. Er zijn op dat front dus geen dringende uitbreidingen nodig. Wel wordt in de verdere toekomst voorzien dat de selectie van te plotten tijdseries uit de diverse subdomeinen beter ondersteund moet worden (gekoppeld aan de bij het vorige punt beschreven hernummering van - in dit geval - uitvoerstations.

Hoofdstuk 4

ogelijkheden van Samenwerking

In het DDPR01 project is ondermeer een koppeling gerealiseerd van de implementatie van domein decompositie binnen CouPle met die binnen Hydra. Een van de lessen die geleerd zijn, is dat de programma's die gekoppeld moeten worden ook algoritmisch voldoende op elkaar moeten aansluiten. Daarnaast zullen er uiteraard voorzieningen moeten bestaan om informatie van het ene programma naar het andere over te brengen.

Verdere samenwerking van WL en RIKZ zal dus afstemming vereisen van zowel de algoritmen die beide instituten gebruiken als van de implementatie van communicatie-software. Op beide punten kunnen RIKZ en WL profiteren van elkaars kennis en ervaring. In dit hoofdstuk zal een en ander nader worden toegelicht.

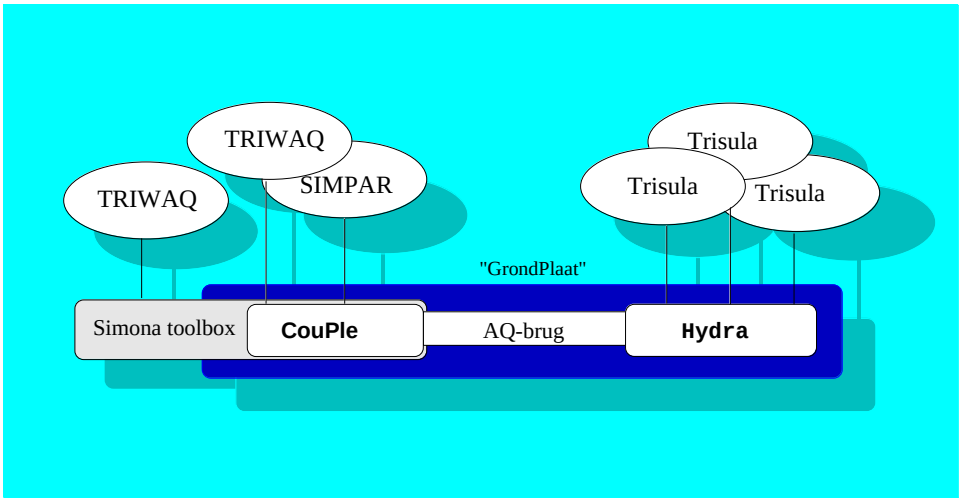
4.1 Algoritmische aspecten

Voor het realiseren van een koppeling is het nodig dat de gebruikte algoritmes op elkaar af te stemmen zijn. Bij TRIWAQ en TRISULA is deze afstemming betrekkelijk eenvoudig omdat beiden een ADI-schema gebruiken en omdat er op het numerieke vlak grote overeenkomsten tussen de twee programma's zijn.

De eerste stap bij het koppelen is het maken van een analyse van beide programma's afzonderlijk. Hierbij moet globaal worden bepaald op welk moment in het algoritme welke informatie nodig is. Bijvoorbeeld het oplossen van de stroomsnelheden (eerste stap in iedere halve tijdstap van het ADI-schema) vereist randvoorwaarden op het nieuwe tijdstip. Het andere programma moet deze informatie kunnen leveren. Anders zijn de programma's niet op elkaar af te stemmen.

In het project is gebleken dat het noodzakelijk is om deze beschrijving van welke informatie op welk moment wordt uitgewisseld volledig te specificeren. Dit gebeurt door middel van een *koppelings-algoritme*. Als voorbeeld van zo'n algoritme is de specificatie voor de TRIWAQ-TRISULA koppeling opgenomen in Bijlage A. Alhoewel de specificatie in eerste instantie zeer globaal kan worden opgezet, moet ze uiteindelijk gedetailleerd en volledig zijn. Aspecten die niet mogen worden vergeten zijn de afstemming van het aantal iteraties dat wordt uitgevoerd en een complete beschrijving van de gegevens die worden uitgewisseld (grootheid, eenheid, grid punten, coördinaten stelsel).

De koppelings-specificatie zorgt voor het onafhankelijk kunnen onderhouden en ontwikkelen van de afzonderlijke programma's. Beide programma's dienen zich te blijven conformeren



Figuur 4.1: *Schematisch overzicht van de wijze waarop de gerealiseerde programmatuur kan dienen als grondplaat voor het koppelen van RIKZ software en WL software.*

aan de specificatie, hetgeen vanzelfsprekend implicaties kan hebben voor de vrijheid om uitbreidingen te plegen. In voorkomende gevallen kan de specificatie worden aangepast. Ook als een derde programma kan voldoen aan de specificatie, dan kan dit in plaats van TRIWAQ of TRISULA gebruikt worden.

4.2 Infrastructuur voor koppelingen

De belangrijkste functie van de concepten Delft-Hydra en CouPle is het bieden van een infrastructuur waarbinnen verschillende koppelingen kunnen worden gerealiseerd. Twee aspecten van deze infrastructuur zijn *proces-management*, het globaal coördineren van rekenprocessen/programma's, en *data uitwisseling*, het converteren en transporteren van gegevens.

In het Delft-Hydra concept wordt de data uitwisseling verzorgd door *Mappers* (aparte processen) en het proces-management door de *Executive*. De mappers en rekenprocessen zijn om beurten actief; de mappers hebben toegang tot de geheugens van de rekenprocessen.

In het CouPle concept verzorgt het *Master* proces het grootste deel van het proces-management, maar niet de evaluatie van stop-criteria. De volledige data uitwisseling wordt verzorgd door communicatie-routines die worden aangeroepen door de reken-processen.

In het project is gebleken dat er grote overeenkomsten tussen deze concepten bestaan. Zo komt het aanroepen van de communicatie-routines in TRIWAQ precies overeen met het om beurten actief zijn van de TRISULA en mapper processen. Verder omvatten de mappers en communicatie routines dezelfde componenten: lezen uit de data-structuur van het rekenproces, eventueel converteren van gegevens (bijv. interpoleren), transporteren van gegevens, en opslaan in de data-structuur van het andere rekenproces. Ook de globale convergentie-checks (proces-coördinatie) worden op vergelijkbare manier afgehandeld. In Delft-Hydra is deze activiteit van de executive op een gedistribueerde manier geïmplementeerd in de mappers. In TRIWAQ wordt hiervoor een communicatie-routine aangeroepen.

De belangrijkste verschillen tussen de concepten zijn het mechanisme van communicatie (met PVM of door middel van shared memory) en de indeling van componenten in processen.

Deze verschillen zijn niet wezenlijk gebleken, en vormden geen hindernis voor het tot stand brengen van de koppeling.

De koppeling is gerealiseerd door een speciaal mapper-proces te maken dat zowel met PVM als via shared memory kan communiceren. Naar TRISULA toe gedraagt deze A/Q-mapper zich precies eender als een normale Delft3dFlow-mapper. Naar TRIWAQ toe gedraagt deze mapper zich als een ander TRIWAQ-proces met communicatie-routines. Dit principe zorgt ervoor dat de koppeling niet ingrijpt op de afzonderlijke omgevingen. Verder zorgden de grote overeenkomsten tussen de concepten ervoor dat deze A/Q-mapper snel gebouwd kon worden. Voor het bouwen van de A/Q-mapper, de "brug" tussen de TRIWAQ/CouPle en de TRISULA/Delft-Hydra omgevingen is als "infrastructuur" in feite alleen gebruik gemaakt van de reeds aanwezige middelen: shared memory access en PVM-communicatie. De beide bestaande omgevingen, uitgebreid met de A/Q-mapper, kunnen tezamen worden gezien als een eerste aanzet tot een grondplaat voor domein decompositie (zie ook Figuur 4.1).

4.3 Conclusies en aanbevelingen

In het project hebben we geen doorslaggevende argumenten kunnen ontdekken voor of tegen een van beide concepten. Beiden hebben hun sterke en zwakke kanten. Ook zijn beide omgevingen ieder op bepaalde punten verder ontwikkeld dan de ander. Het WL heeft bijvoorbeeld routines voor het converteren van randvergelijkingen van het ene rekenrooster naar het andere. Daartegenover heeft RIKZ programmatuur voor onregelmatige rooster-opdelingen en voor het communiceren tussen computers zonder shared memory. Het Delft-Hydra concept lijkt iets geschikter voor dynamisch proces-beheer. Het CouPle concept lijkt iets eenvoudiger te zijn en stelt minder eisen aan het computer platform.

Omdat beide concepten hun voordelen hebben is het verstandig om ze naast en met elkaar te laten voortbestaan. Dit is mogelijk omdat de koppeling van de beide omgevingen niet veel inspanning vereist en niet ingrijpt op de concepten.

In feite bieden beide concepten een vorm van interproces communicatie en proces coördinatie. Functioneel liggen beide concepten dicht bij elkaar en het lijkt dan ook mogelijk om een set van basis-operaties te definiëren die in beide concepten terug te vinden zijn en die met elkaar voldoende zijn om communicatie tussen en coördinatie van een set van processen te bouwen. Die set van basis-operaties zal dan in ieder geval een `send`- en `receive`-operatie en een globale reductie operatie moeten omvatten, alsmede routines voor interpolatie.

Deze basis-componenten kunnen van elkaar worden overgenomen of samen worden ontwikkeld. Bij het overnemen van componenten vormen de verschillende implementaties wel enigszins een hindernis. Dit kan worden vergemakkelijkt door de concepten op gedetailleerder niveau (met name de interne administraties) op elkaar af te stemmen.

Om beide concepten optimaal op elkaar af te stemmen is een aantal acties op de korte termijn wenselijk:

- het regelmatig plegen van overleg over de ontwikkeling van beide concepten;
- het inventariseren, documenteren en waar mogelijk stroomlijnen van de gehanteerde interne administraties; dit kan goed gebeuren door het gezamenlijk oppakken van een tweetal ontwikkelingen die in beide instituten gepland zijn, te weten:
  - de ontwikkeling van m:n koppeling in het horizontale vlak

  - het ontwikkelen van voorzieningen voor communicatie tussen en coördinatie van processen met verschillende tijdstappen

- het uitwisselen van kennis en/of programmatuur op het gebied van parallel rekenen (VORtech Computing, RIKZ → WL) en domein decompositie (WL → RIKZ). Op dit vlak liggen er met name mogelijkheden op het gebied van
  - preprocessing: partitioneringskennis van RIKZ en kennis van roostergeneratie van WL;
  - koppelingsvergelijkingen, waarin met name het WL sterk is;
  - parallel rekenen, waarin RIKZ een voorsprong heeft.

Referenties

[1] B.M. Gerritsen, W. Zijl, F.J.T. Floris, and J.L. van der Meij. Invoering Domein-Decompositie RIKZ. Technical Report GG R 96-76 C, TNO Grondwater en Geo-Energie, Postbus 6012, 2600 JA DELFT, december 1996. In opdracht van RIKZ.

[2] H.X. Lin, H.H. ten Cate, M.R.T. Roest, and E.A.H. Vollebregt. Technical documentation online coupling of SIMPAR and TRIWAQ. Technical Report 1.1, Delft University of Technology, Delft University of Technology, ITS/TA/WAGM, Postbus 5031, 2600 GA Delft, July 1997.

Bijlage A

Specificatie van het koppelings-algoritme voor TRI A -TRISULA

Als voorbeeld van een koppelings-algoritme geven we in Figuur A.1 de specificatie die is gebruikt voor koppeling van TRIWAQ en TRISULA, waarbij beiden een ADI-schema gebruiken. De specificatie omvat alleen het stromings-gedeelte. Het koppelings-algoritme is groot omdat op gedetailleerd niveau wordt ingegrepen in beide codes. De specificatie van de gegevens die worden uitgewisseld is daarentegen beperkt omdat TRIWAQ en TRISULA hierin sterk op elkaar lijken en omdat ervan wordt uitgegaan dat de roosters horizontaal aansluiten.

De stappen tussen haakjes geven de echte berekeningen weer in beide programmas. Ze staan tussen haakjes omdat het voor de koppeling niet heel veel uit maakt hoe ze worden geïmplementeerd.

Zowel TRIWAQ als TRISULA gebruiken de conventie dat een grid-punt (array-index) bestaat uit een `Zeta`-punt, het `U`-snelheids-punt RECHTS hiervan en het `V`-snelheids-punt er BOVEN:

```
          - - - - - - - - -
  ^      | V   | V   | V   |
  | N /  |     |     |     |
  | eta  | Z U | Z U | Z U |
          - - - - - - - - -

              ---> toenemende M/Xi-coordinaat
```

Als we een interface/subdomein-rand hebben door `U`-punten `(m,n0) ... (m,n1)` (verticale interface) dan worden de variabelen gecommuniceerd voor de volgende verzamelingen van gridpunten:

```
Zeta        (stappen 1, 18, 41) : L->R : (m  ,n0-1) .. (m  ,n1+1)
                                : R->L : (m+1,n0-1) .. (m+1,n1+1)
U           (stappen 1, 23, 32) : L->R : (m-1,n0-1) .. (m-1,n1+1)
                                : R->L : (m+1,n0-1) .. (m+1,n1+1)
```

```
Intitialisatie:

1)      Communicate Kmax, Sigmas
2)      Communicate Zeta, U, V

Doe voor alle tijdstappen:

3)      For all timesteps:

        Eerste halve tijdstap:

4)              Communicate W, VERT-VISCOS
5)              (Build system of eqs for V)
6)              iter=0
7)              While (NOT converged AND iter<MAX IT 1)
8)                      iter=iter+1
9)                      (Solve for V, determine flag LocalConv)
10)                     Communicate V
11)                     Communicate convergence flags
12)             End While
13)             While points dried
14)                     (Build eqs for Zeta/U)
15)                     iter=0
16)                     While (NOT converged AND iter<MAX IT 2)
17)                             iter=iter+1
18)                             (Solve for Zeta, determine flags LocalConv, pointsDried)
19)                             Communicate Zeta
20)                             Communicate convergence flags
21)                     End While
22)             End While
23)             (Solve for U)
24)             Communicate U
25)             (Calculate W)
26)             (Calculate VERT-VISCOS)

        Tweede halve tijdstap:

27)             Communicate W, VERT-VISCOS
28)             (Build system of eqs for U)
29)             iter=0
30)             While (NOT converged AND iter<MAX IT 1)
31)                     iter=iter+1
32)                     (Solve for U, determine flag LocalConv)
33)                     Communicate U
34)                     Communicate convergence flags
35)             End While
36)             While points dried
37)                     (Build eqs for Zeta/V)
38)                     iter=0
39)                     While (NOT converged AND iter<MAX IT 2)
40)                             iter=iter+1
41)                             (Solve for Zeta, determine flags LocalConv, pointsDried)
42)                             Communicate Zeta
43)                             Communicate convergence flags
44)                     End While
45)             End While
46)             (Solve for V)
47)             Communicate V
48)             (Calculate W)
49)             (Calculate VERT-VISCOS)

        einde van tijdstap

50)     End For
```

Figuur A.1: *Koppelings algoritme voor de TRISULA-TRIWAQ koppeling*

```
V           (stappen 1,  9, 46) : L->R : (m  ,n0-1) .. (m  ,n1+1)
                                : R->L : (m+1,n0-1) .. (m+1,n1+1)
W           (stappen 3, 26)     : zelfde als Zeta
VERT-VISCOS (stappen 3, 26)     : zelfde als Zeta
```

Opmerkingen:

- Het is in TRIWAQ zo dat `W` en `VERT-VISCOS` zijn gedefinieerd in dezelfde punten (hor.) als `Zeta`.
- Bovenstaande specificatie wordt in alle gevallen gebruikt, ook bijvoorbeeld als de genoemde grid-punten op een gesloten rand liggen. Eventueel mag in zulke gevallen een dummy-waarde worden ingevoegd in plaats van een echte waterstand / snelheid / .. . Hiervan moet wel melding worden gemaakt.
- Bovenstaande specificatie is "ruim" in de zin dat ze voldoende is voor veel verschillende configuraties van domeinen en algoritmes. Verder is ze eenvoudig omdat telkens de `n`-coördinaat van `n0-1` tot `n1+1` loopt. Echter, informatie voor sommige punten zal veelal overbodig zijn, bijv `U` voor hoekpunten `(m-1,n0-1)` en `(m+1,n1+1)`, en `V` voor `(m,n1+1)`.
- de informatie is niet noodzakelijk voor alle gridpunten die hierboven staan genoemd, zeker niet in de tryout.

Als we een interface/subdomein-rand hebben door `V`-punten `(m0,n) ... (m1,n)` (horizontale interface) dan worden de variabelen gecommuniceerd voor de volgende verzamelingen van gridpunten:

```
Zeta        (stappen 1, 18, 41) : O->B : (m0-1,n  ) .. (m1+1,n  )
                                : B->O : (m0-1,n+1) .. (m1+1,n+1)
U           (stappen 1, 23, 32) : O->B : (m0-1,n-1) .. (m1+1,n-1)
                                : B->O : (m0-1,n+1) .. (m1+1,n+1)
V           (stappen 1,  9, 46) : O->B : (m0-1,n  ) .. (m1+1,n  )
                                : B->O : (m0-1,n+1) .. (m1+1,n+1)
W           (stappen 3, 26)     : zelfde als Zeta
VERT-VISCOS (stappen 3, 26)     : zelfde als Zeta
```