Chapter 11
Model Coupling with OpenMI Introduction of Basic Concepts

Bernhard Becker and Andreas Burzel

Abstract Interaction processes between two or more model domains can be represented with the help of model coupling. Different methods of coupling apply for different interaction processes. We illustrate this with the help of an exercise. In order to facilitate model coupling of water-related models the OpenMI standard has been developed. This document gives an introduction to the open modelling interface (OpenMI) and explains the steps that are necessary to migrate existing model code to Open MI compliance. An OpenMI composition of a flow simulation model for a river section of the Elbe river (Germany) that is coupled with a model for the control of a hydraulic structure is used to explain how models can be coupled with Open MI and to illustrate the added value of model coupling in terms of improved simulation result.

1 Introduction

This document is accompanying course material for trainings that have been given within the frame of the CIPRNet project (www.ciprnet.eu). The first objective of this document is to provide a general introduction into model coupling. Learning goal is to know basics of different coupling methods and different modes of process interaction modelling. The second section provides technical explanation of the OpenMI standard. Students learn what an OpenMI compliant component is and learn how the data exchange works. The third section accompanies the OpenMI life demonstration, where two OpenMI-compliant models are loaded and connections between models are configured. This document also contains a reference list for further reading.

B. Becker (✉) · A. Burzel
Deltares, P.O. Box 177, 2600 MH Delft, The Netherlands
e-mail: bernhard.becker@deltares.nl

A. Burzel
e-mail: andreas.burzel@deltares.nl

R. Setola et al. (eds.), *Managing the Complexity of Critical Infrastructures*,
Studies in Systems, Decision and Control 90,
DOI 10.1007/978-3-319-51043-9_11

2 Model Coupling and Conjunctive Modelling

2.1 What Is a Model?

> A model should be made as simple as possible, but not simpler.
>
> (after Albert Einstein, 1879–1955)

Following Konikow and Bredehoeft [1] we use the following definitions:

A *model* is a representation of a real system or process. A *conceptual model* is a hypothesis for how a system or process operates. *Mathematical models* are abstractions that replace objects, forces, and events by expressions that contain mathematical variables, parameters and constants. *Deterministic models*, also called physics-based models, are based on the conservation of mass, momentum and energy. Deterministic models often require the solution of differential equations for certain boundary and initial conditions. A mathematical model, or, more in particular, a numerical algorithm to solve differential equations, implemented into computer code is called a *computer model*. This computer model can also be considered as a *generic model*. When model parameters, boundary conditions and grid definitions for a generic model are specified to represent a particular geographic area, we obtain a *site-specific model*, including model data and software. A *synthetic model* represents a fictitious site, often used to illustrate or analyse a certain process.

A computer model usually consists of a graphical user interface part and a computational core that solves the partial differential equation system.

Flow processes are often described mathematically by partial differential equations. These equations cannot be solved analytically. The numerical solution requires a grid (mesh) that represents the modelling area. The solution of differential flow equations requires a full definition of the boundary of the modeling area, the so-called boundary conditions. In addition, internal boundary conditions like sources and sinks can be defined. Transient flow problems require initial conditions for the whole modeling area grid. A set of boundary conditions and initial conditions is called *scenario*.

2.2 What Is Conjunctive Modelling?

Conjunctive modelling means to link site-specific models in such a way that the interaction processes between the domains the models represent are modelled on a time-step basis. There are different levels of conjunctive modelling: model coupling means data transfer in two directions, while an uncoupled approach has data exchange in one direction only.

If models are coupled, the simulation results of the first model have an impact on the second model and vice versa. This means that coupled models must exchange data during runtime on a time step basis. In case of uncoupled conjunctive

modelling the simulation results of the first model have an impact on the second one, but the simulation result of the second model has no feedback impact on the first model.

According to Morita and Yen [2, 3], there are three levels of model coupling:

- simultaneous coupling
- alternating iterative coupling
- externally coupling.

External coupling means data exchange once per time step in both directions. Results from one model are used as boundary conditions in the other one and vice versa (see Fig. 1a). This is the lowest level of model coupling. Also called time-lagged approach [4, 5] this approach is the least accurate one, because it contains inherent mass balance and momentum balance errors. But this approach is certainly the most often applied one, because it is easier to implement than the other two, and often sufficient.

Iterative coupling means to exchange data between models not only once per time step, but to iterate the exchange of data until a certain convergence criterion is achieved (see Fig. 1b). Consequently, mass balance errors and momentum errors are basically smaller than for external coupling. But this method is more difficult to implement and more computational expensive.

Simultaneous coupling is the highest level of model coupling. It means to represent different processes, including the interactions, in one equation system. However, the simultaneous solution requires equal time stepping for all coupled processes, and the equations should be of the same type to make it efficient.

OpenMI supports iterative coupling and external coupling. Morita and Yen [2] and Becker and Talsma [6] discuss numerical aspects of these model coupling approaches.

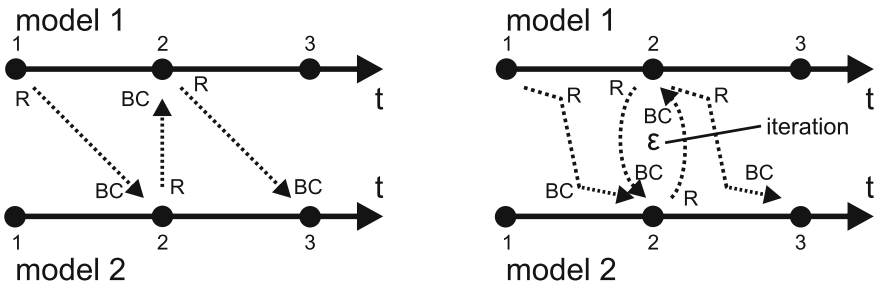


Fig. 1 Functional principle of external coupling and iterative coupling of two models (after Becker [28]). *R* result, *BC* boundary condition, *t* time, ε convergence criterion

As uncoupled approach we consider the successive execution of two model simulations where the first model produces boundary conditions for the second one. A feedback from the second one to the first is not incorporated. An uncoupled conjunctive modelling can be realized by simple exchange of input and output files between models. The easiest way to implement such an uncoupled conjunctive modelling is to implement simulation results from one model as boundary conditions for the second model manually or script-based. OpenMI can help to improve efficiency for uncoupled conjunctive modelling as shown by Becker and Schüttrumpf [7]. More advanced approaches of uncoupled conjunctive modelling incorporate a data integration platform like Deft-FEWS [8].

2.3 Task

Your task is the design of a model chain for the following scenario:

1. Heavy rainfall causes high water in a river.
2. High water in a river causes dike breach due to overtopping.
3. The dike breach causes inundations of the hinterland.
4. From the inundated areas water infiltrates into the subsurface and causes groundwater head rise.
5. Rising groundwater levels create uplift forces on a road tunnel and flows cellars with information technology installation.

Carry out the following working steps:

1. Identify the relevant processes and the corresponding models.
2. Draw a flow chart with the models and their interactions. Indicate the direction of data transfer with arrows.
3. Explain your model chain.
4. Discuss alternative set-ups.

A possible solution of task 1 is given in Table 1. A solution for task 2 is given in Fig. 2. A possible explanation of the model chain (task 3) is

Table 1 Relevant processes and corresponding models

No.	Process	Model
1	Rainfall-runoff	Hydrological model
2	River flow	1D open channel flow model
3	Dike breach	Dike breach model
4	Hinterland flooding	Two-dimensional flood model
5	Groundwater head rise (subsurface flood)	Groundwater model

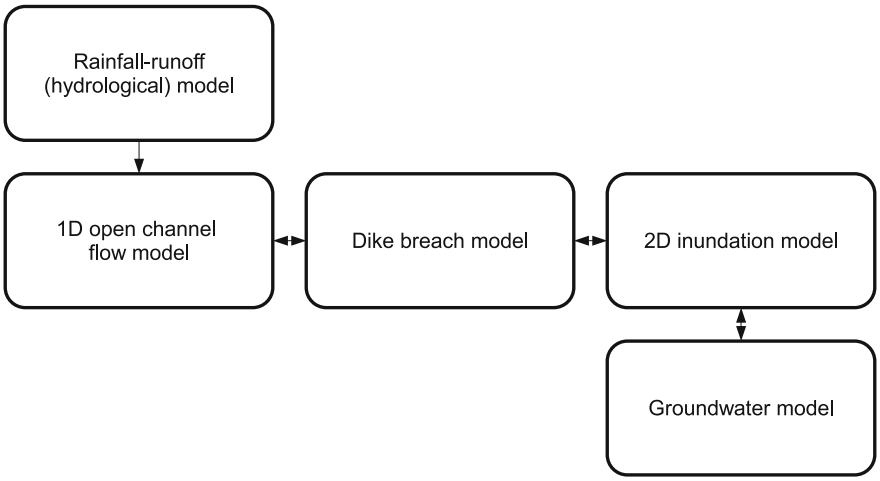


Fig. 2 Model coupling for process interaction modelling. *Arrows* indicate data exchange between models to represent process interaction

- Rainfall runoff feeds the open channel flow, but the open channel flow processes have no impact on the rainfall-runoff. So the data transfer is unidirectional and the interactions can be modeled uncoupled.
- River flow, dike breach and inundation are processes that interact with each other. Uncoupled modelling would violate the mass balance of water, so a coupled approach is chosen.
- The infiltration of water from inundated areas into groundwater is an interaction process which cannot be modelled uncoupled, because infiltrating water affects the inundation area and the groundwater balance.
- The model chain provides information that can be used to identify endangered critical infrastructure. The infrastructure itself has no impact on the hydrological processes, so the simulation results can be transferred to critical infrastructure models manually.

Alternative setups (task 4):

- A connection between the river model and the groundwater model adds the process of bank storage to the system model.
- Interactions between river model, dike breach model and two-dimensional flow model could be made uni-directional to trade-off accuracy against performance.
- A geotechnical model for failure mechanisms due to uplift forces can be added to the modelling chain.

3 The OpenMI Standard

3.1 Introduction

The OpenMI standard defines an interface that allows time dependent models to exchange data at runtime [9]. Model components that comply with the OpenMI standard can, without any programming, be coupled to OpenMI modelling systems [10, 11]. The OpenMI environment provides tools that facilitate the migration of legacy code. This grants a high acceptance of coupled models by users, because they can use their already existing models in coupled simulations. The initiative for OpenMI originates from the water sector, but OpenMI has already reached a wider distribution than the water domain only (see e.g. Bulatewicz et al. [12]).

Beside the standard interface specification, the OpenMI-association [13] also provides the OpenMI environment. This is a software that assists in the implementation of the OpenMI standard. It contains compiled .NET assemblies and the source code of all packages and their documentation [9]. The OpenMI environment also provides the OpenMI configuration editor. This programme supports the data exchange between different OpenMI compliant components.

An OpenMI system is a software system where different OpenMI compliant components are connected to a coupled modelling system. The OpenMI data exchange is based on a pull-driven request-reply mechanism. One component, for example a site-specific model, requests data needed for the own computation from another component. Components can be connected in different manners:

- unidirectional connection
- bidirectional connection
- iterated connection.

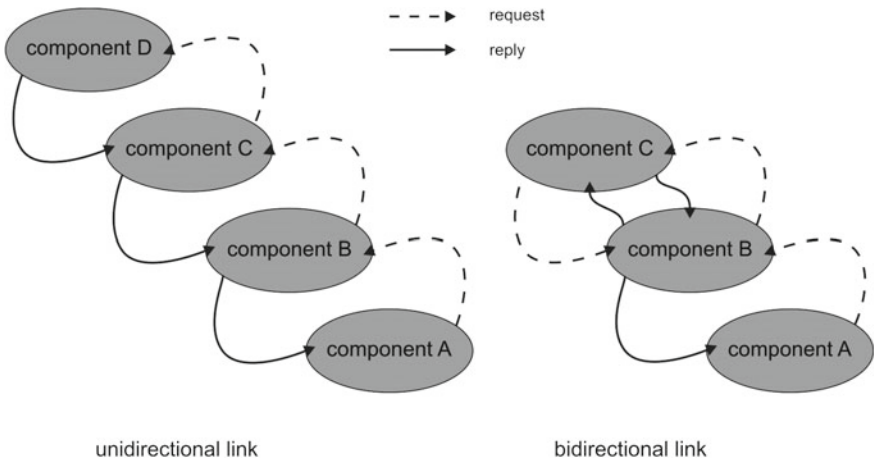


Fig. 3 Different connection layouts with the request-reply mechanism (after Gregersen et al. [10])

According to Sect. 2.2, unidirectional connection supports the uncoupled approach, the bidirectional connection is for external coupling and the iterated connection helps to realize an iterated coupling with OpenMI. The simultaneous solution cannot be achieved with OpenMI.

In Fig. 3, different layouts of pull-driven request-reply connections are shown. For the unidirectional chain, component A requests B for data. In order to response, it needs data from another component itself and requests C for data, which again requests data from component D. D is at the end of the chain and performs its computation first and then answers C. C is now able to compute and answers the request of component B afterwards. B now calculates with the data from C and is able to respond on the request of A. For the bidirectional connection example, component A requests data from B. B needs data from component C. To fulfil this request, C needs data from B. Because B waits for data from C itself, it gives a guess to C. C computes with this guess and can now response to B. B is now able to compute and to reply to the request of A.

Both examples show, that one component must initialize the computation with a request to define which component shall compute first. That is why each OpenMI system contains an element which triggers the simulation. For the bidirectional connection, simulation results may differ depending on which component computes first and gives a guess. Gregersen et al. [10] call this coupling semi-explicit, because the results of one component are based on a guess, but the results of the other component are based on a calculation. The iterative connection is an advanced bidirectional connection. In the example of Fig. 3 (right side), components B and C would adjust their reply values iteratively until an accuracy criterion is fulfilled.

3.2 *OpenMI Composition Components*

The `omi`-file contains information about one single OpenMI compliant component:

Table 2 `omi`-file for a SOBEK model

```
<?xml version="1.0"?>
<LinkableComponent xmlns="http://www.openmi.org"
  Type="DeltaShell.OpenMIWrapper.DeltaShellOpenMILinkableComponent"
    Assembly=".\bin\DeltaShell.OpenMIWrapper.dll">
  <Arguments>
    <Argument Key="DsProjFilePath" ReadOnly="true"
      Value=".\SobekRiverFlowModel\Magdeburg3.dsproj" />
    <Argument Key="ModelName" ReadOnly="true" Value="integrated model" />
    <Argument Key="ResultingDsProjFilePath" ReadOnly="true"
      Value=".\SobekRiverFlowModel\MagdeburgRTC.dsproj" />
    <Argument Key="SplitSpecificElementSets" ReadOnly="true"
      Value="CalcPoints;Laterals;Structures;Measurements" />
  </Arguments>
</LinkableComponent>
```

- Where is the DLL with the computational core and OpenMI-Interface?
- Where is the working directory with input files?
- Anything else like command line arguments or specific settings?

The omi-files are structured in xml. The omi-file must be created by the modeller. An example of an omi-file is given in Table 2.

Table 3 opr-file for an OpenMI composition with a SOBEK model and an RTC-Tools model

```
<guiComposition version="1.0">
  <models>
    <model omi="d:\OpenMIcourse\OpenMICoursePackageElbe\RtcTools.omi"
      rect_x="195" rect_y="113" rect_width="100" rect_height="51" />
    <model omi="d:\OpenMIcourse\OpenMICoursePackageElbe\ElbeSobek.omi"
      rect_x="52" rect_y="112" rect_width="100" rect_height="51" />
    <model omi="Oatc.OpenMI.Gui.Trigger"
      rect_x="196" rect_y="30" rect_width="100" rect_height="51" />
  </models>
  <links>
    <uilink
      model_providing="integrated model"
      model_accepting="RtcTools_ModelId">
      <link id="2"
        source_elementset="ObservationPoint1"
        source_quantity="Water level (op)"
        target_elementset="Water level (op)@ObservationPoint1"
        target_quantity="Water level (op)" />
    </uilink>
    <uilink
      model_providing="RtcTools_ModelId"
      model_accepting="integrated model">
      <link id="4"
        source_elementset="Crest level (s)@Weir1"
        source_quantity="Crest level (s)"
        target_elementset="Weir1"
        target_quantity="Crest level (s)" />
    </uilink>
    <uilink
      model_providing="integrated model"
      model_accepting="Oatc.OpenMI.Gui.Trigger">
      <link id="6"
        source_elementset="Node001"
        source_quantity="water_level"
        target_elementset="TriggerElementID"
        target_quantity="TriggerQuantityID" />
    </uilink>
  </links>
  <runproperties
    listenedeventtypes="11111111111"
    triggerinvoke="01/26/2000 00:00:00"
    runinsamethread="0" showeventsinlistbox="1"
    logfilename="CompositionRun.log" />
  <mainForm width="360" height="277" />
  <sdk>
    <smartbuffer maxnumberoftimes="0" />
  </sdk>
</guiComposition>
```

3.3 Connections

The opr-file defines how OpenMI components are connected within an OpenMI composition and contains runtime information:

- Which components are part of the composition (reference to omi-files and trigger component)?
- Which connections are defined between components?
- Details of the connections (what and where)?
- Simulation period.

The opr-file is created by the OpenMI configuration editor, but can be modified by the modeller. Like the omi-file, the opr-file is structured in xml. Table 3 gives an example for an opr-file.

A connection between model components consists of links. A link is defined between an output exchange item and an input exchange item of two different model components, respectively. An exchange item defines a simulation time related quantity and its unit for an ordered set of elements, e.g. a single node number, a node coordinate, or lines, polygons or polyhedrons. Input exchange items usually form boundary conditions in an OpenMI compliant model component, while output exchange items are mostly simulation results.

During simulation, the exchange item is assigned with a value. This gives the OpenMI compliant component the following information:

- the *value* itself,
- *what* the value represents (quantity and unit),
- *where* the value applies (element set),
- and *when* the value applies.

The OpenMI compliant component is responsible to provide the data in a correct way and for what to do with received data.

3.4 Making (Legacy) Code OpenMI Compliant

An OpenMI-compliant model satisfies the following criteria:

1. The model must be able to submit to run-time control by an outside entity.
2. The model must be structured in such a way that initialization is separate from computation.
3. The model must be able to expose information on the modelled quantities it can provide.
4. The model must be able to provide the values of the modelled quantities for any requested point in time and space.

5. The model must be able to respond to a request; if the response requires data from another component, the model must be able to pass on the time in its own request.
6. A delivering model component must know what time it has reached. It must recognize whether it has not yet reached the requested time, it is at the requested time or it has passed the requested time.
7. Components must be able to interpolate if the requested time is not in their own time step or space frame.
8. Components must know when they are waiting for data, and in which case they will have to return an extrapolated value.

Since OpenMI is an interface standard, the implementation of the interface requires modifications of the source code of a mode component that shall run within OpenMI compositions. The easiest way to make a generic model OpenMI-compliant is to embed the code into a wrapper class provided by the OpenMI environment [14]. Therefore, the code usually has to be reorganized. The wrapper controls the run-time activity of pulling data across links. The OpenMI environment provides a "smart wrapper" that already handles most of the tedious and difficult tasks to be performed, for example items 3–8 from the list above.

An OpenMI compliant model component is loaded into the OpenMI configuration editor as dynamic link library (DLL). To comply with the OpenMI standard, a component must provide several functions (OpenMI methods). Examples for those methods concerning the structure of the programme are listed below [14]:

1. `Initialize()`
2. `PerformTimeStep()`
3. `Finish()`
4. `Dispose()`

The method `Initialize` usually comprises the opening and reading of input files describing the mesh, initial conditions and boundary conditions. `PerformTimeStep` initializes the computation of one time step. The `Finish` method has been prepared to close all files used by the component; within the `Dispose` method, allocated memory is freed. The most important OpenMI methods for the data exchange itself are given in the following list:

- `GetCurrentTime()`
- `GetValues(QuantityID, ElementSetID)`
- `SetValues(QuantityID, ElementSetID, values)`

`GetCurrentTime` returns the point in time a component has reached. `GetValues` returns values related to output exchange items (simulation results) for the current time. The function arguments indicate what the return value represents and where it is located. The `SetValues` method sets a value for the model component as an answer on a request. The value to set is a function argument and is usually used as a boundary condition value by the model.

3.5 Example Cases of Conjunctive Modelling with OpenMI

Example cases of conjunctive modelling with OpenMI under contribution of the authors of this document are given in the following list:

- Generation of boundary conditions for a transient dam seepage scenario [7].
- Modelling of surface-subsurface interactions, i.e. bank storage and vertical infiltration from a flooded area [15].
- Coupling of an open channel flow model with a pump model to design a large pump station [16].
- Coupling of models of the same type: two open channel flow models are coupled to bridge administrative boundaries [17, 18].
- Integration of different hydrological processes [19].
- Real-time control of hydraulic structures in open channel flow models to model the human interactions in a water system [20–23].

See also the OpenMI website www.openmi.org for more publications.

4 Example: Coupled Flow Simulation and Control

4.1 Study Area and Modelling Objective

The study area is a part of the Elbe river at Magdeburg (Germany). An overview of the study area is given in Fig. 4. The modelling objective is to manage the river in such a way that the water levels remain below the flood warning level. Beside the city of Magdeburg, the critical infrastructure

- main station and
- two railway junctions

might be affected in case of flooding.

4.2 Approach

The relevant processes are

- open channel flow in the section of the river Elbe and
- human operations in the river system (control of hydraulic structures).

We use two models to represent these processes:

- a SOBEK open channel flow model for the flow of water in the Elbe river and
- a real-time control model RTC-Tools to represent the human operations in the water system.

Fig. 4 Study area (taken from www.maps.google.com)

4.3 The SOBEK Open Channel Flow Model

The SOBEK open channel flow model is a deterministic model that simulates water flow in rivers by solving the Saint-Vanant equations with the so-called staggered grid numerical scheme [24].

The SOBEK schematization "Elbe at Magdeburg" is shown in Fig. 5. The water system model network has the following characteristics:

- one branch in the south
- one branch in the north
- two branches in the centre, one representing the main river and one represents the Old Elbe branch

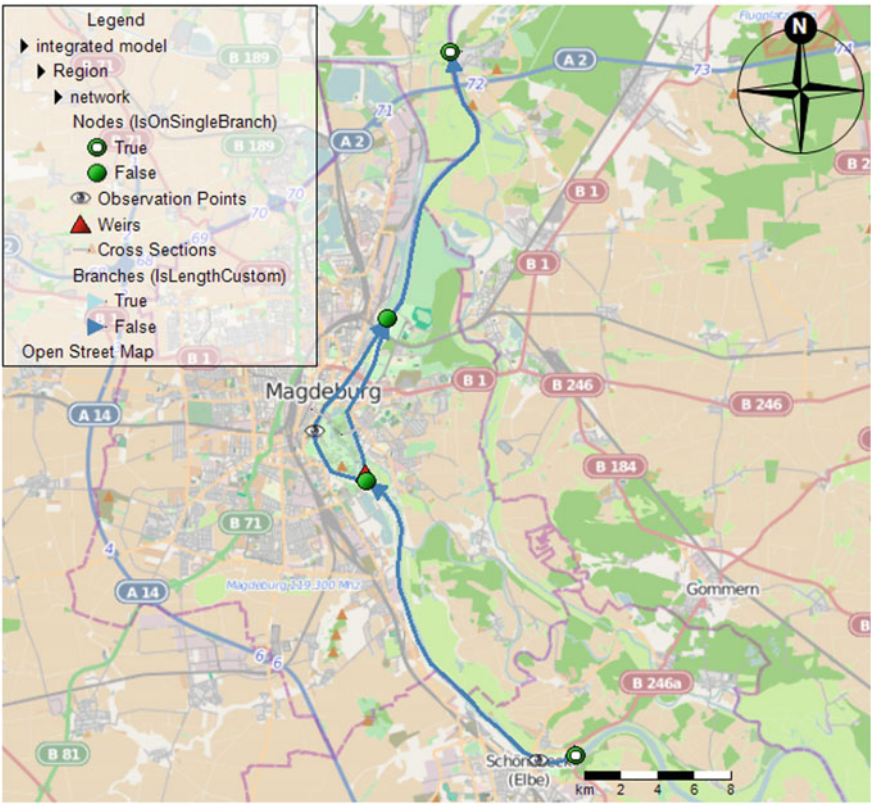


Fig. 5 SOBEK open channel flow model network. Water flows from south to north (background map from www.openstreetmap.org)

- cross sections
- observation points
- one weir to close the Old Elbe branch on its upstream end.

The upstream boundary condition is a discharge time series, as downstream boundary condition a rating curve (discharge-water level relation) is set.

Task:

- Open the SOBEK model.
- Inspect the network: find the observation points and the structure.
- Look at the inflow boundary.
- Run the model.
- Inspect the side-view for the two routes “Elbe” and “Old Elbe”.
- Look at the hydrographs for the two observation points.

For modelling with SOBEK see the user manual [25].

4.4 The RTC-Tools Real-Time Control Model

The RTC-Tools model [22, 26] addresses the control of the weir which is represented in the SOBEK model as structure node. The control is based on water level observations at the Schönebeck gauge in the upstream part of the model. The gauge is represented in the SOBEK model as observation point. The control flow is given in Fig. 6 as a decision tree. This RTC-Tools model is not a deterministic model, but belongs to the group of logical models.

A trigger evaluates if the observed water level at Schönebeck is greater than 54 m. If the condition is true, the weir is opened, if not, the weir is closed. This simple operational protocol ensures sufficient water depth for cargo ship navigation in the main channel of the Elbe during normal condition and reduces the water level during high water conditions.

Task:

- Open the file `rtcToolsConfig.xml`.
- Find the trigger and rule elements from the flow chart in Fig. 6.

See the manual [27] for details on working with RTC-Tools.

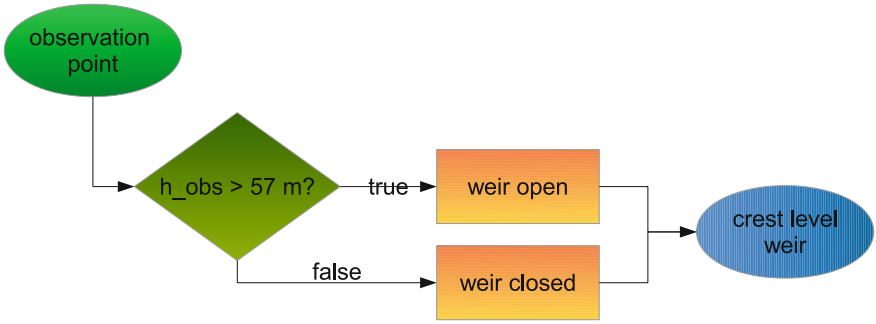


Fig. 6 Flow chart for the control of the weir as modelled with RTC-Tools

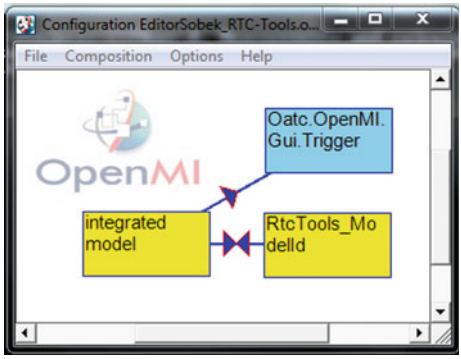


Fig. 7 OpenMI Configuration Editor with a SOBEK model component, an RTC-Tools model component and an OpenMI trigger component

4.5 Coupling with OpenMI

The interaction of human control and open channel flow is as follows:

- the crest level of the weir is controlled in dependence of the current water level at the observation point and
- the control of the weir has an impact on the water system:
  - if the weir is open, water can flow through the main branch and the Old Elbe branch
  - if the weir is closed, the water flows through the main Elbe branch only.

To model this interaction, bi-directional data exchange has to be configured as follows:

- SOBEK provides the water level at Schönebeck gauge to RTC-Tools
- RTC-Tools provides the crest level for the weir to SOBEK.

Task:

- Open the OpenMI configuration editor.
- Load the RTC-Tools model into the OpenMI configuration editor.
- Load the SOBEK model into the OpenMI configuration editor.
- Add a trigger component to the composition. Note that the OpenMI trigger should not be confused with the RTC-Tools trigger element.
- Add a connection from the RTC-Tools model to the SOBEK model and configure the connection as shown in Fig. 8.
- Add a connection from the SOBEK model to the RTC-Tools model and configure the connection as shown in Fig. 9.

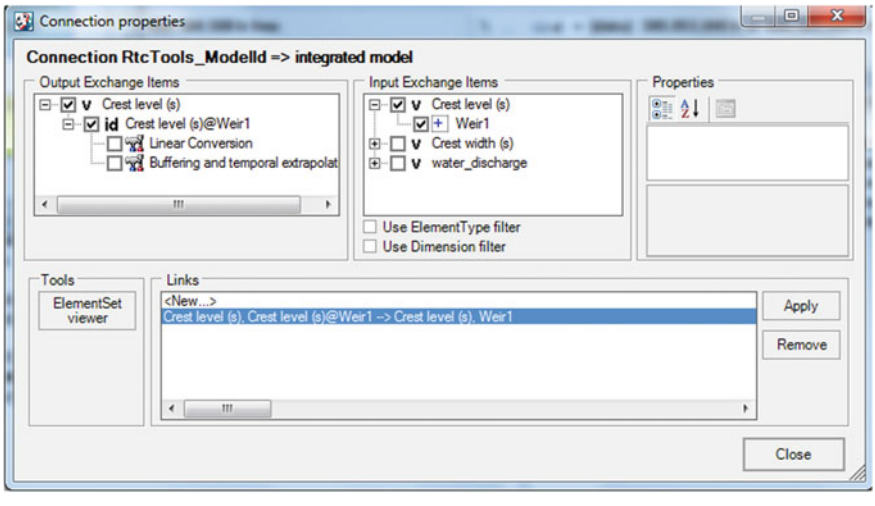


Fig. 8 Connection properties RTC-Tools—SOBEK

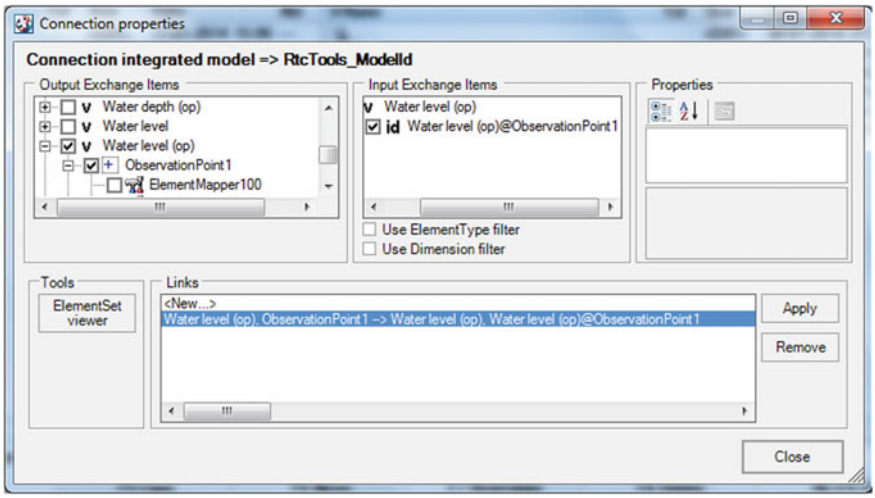


Fig. 9 Connection properties SOBEK—RTC-Tools

- Add a connection from the SOBEK model to the OpenMI trigger and configure the connection as shown in Fig. 10. Choose an arbitrary exchange item from the SOBEK model.
- Save the composition. The OpenMI composition should look like the one in Fig. 7.

The functional principle of the data exchange is shown in Fig. 11. The data exchange procedure can be summarized as follows [6]:

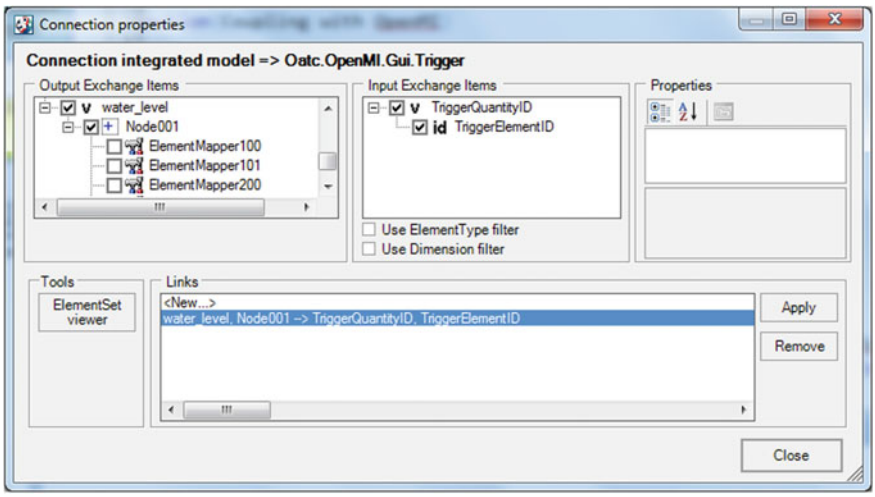


Fig. 10 Connection properties SOBEK—OpenMI trigger

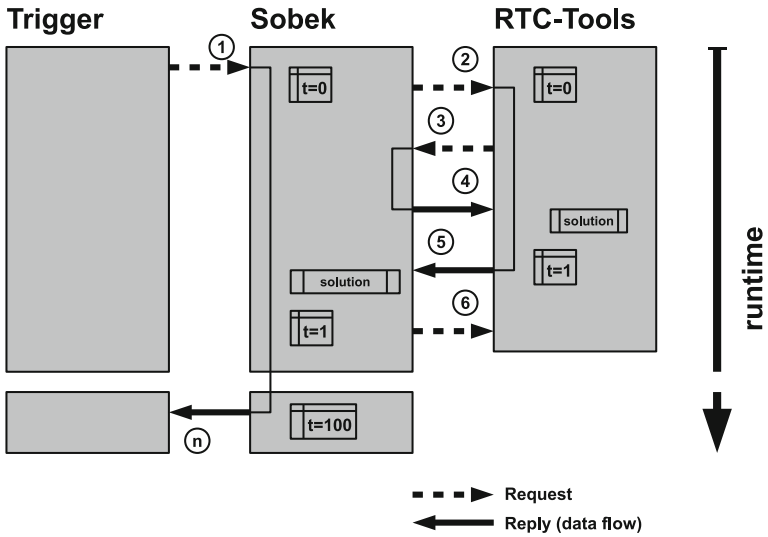


Fig. 11 Request-reply mechanism for an OpenMI composition with SOBEK and RTC-Tools

- The model component that asks first computes last.
- The model that asks gives the guess (i.e. data from the previous time step).

The OpenMI trigger element has been connected to the model component in such a way that RTC-Tools is the first model that computes the solution for a given time step. In order to compute the control action for the current time step, RTC-Tools uses observed data from SOBEK from the previous time step. This time lag (see also Sect. 2.2) is usually a source of inaccuracy when coupling physical processes, but in the current case it ensures that a control action takes effect in the water system *after* the observation that triggers the control action has been made.

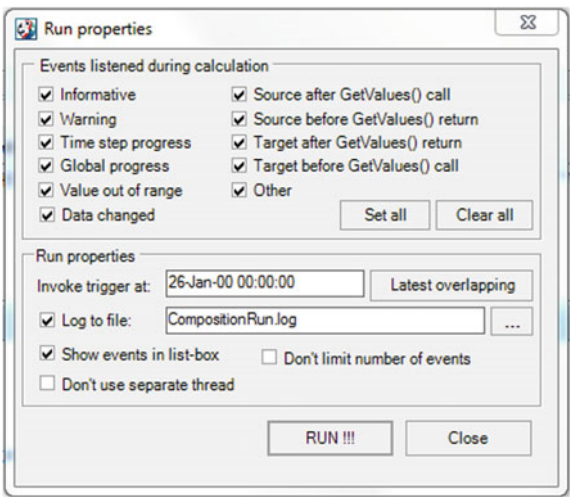


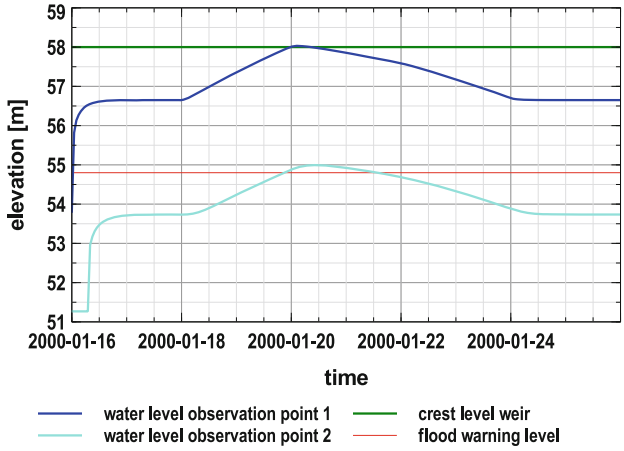
Fig. 12 OpenMI configuration editor Run properties window

4.6 Coupled Simulation and Simulation Results

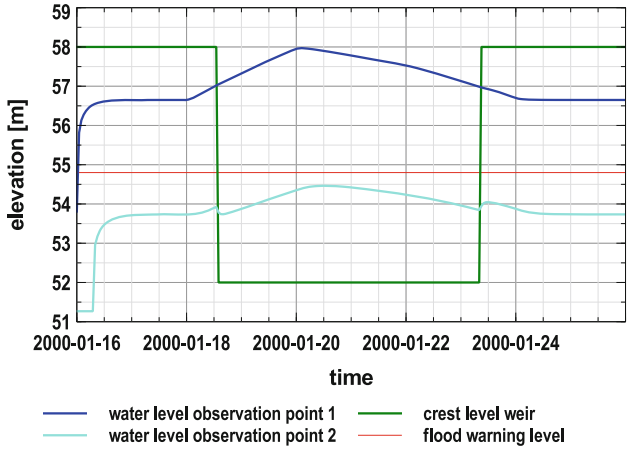
Task:

- Run the OpenMI composition via the Run properties window (Fig. 12).
- Open the SOBEK model that has been running within the OpenMI coupled simulation.
- Inspect the side views for the routes “Elbe” and “Old Elbe”. For the latter one, add the coverage “Crest level(s)”.
- Inspect the hydrographs of the two observation points and the crest level.

Figure 13 shows simulation results from the SOBEK model with an uncontrolled weir (Fig. 13a) and the simulation results from the coupled simulation



(a) Simulation results from the SOBEK model (channel flow)



(b) Simulation results from the coupled flow simulation with SOBEK (channel flow) and RTC-Tools (control)

Fig. 13 Simulation results

SOBEK—RTC-Tools (Fig. 13b), where the weir is controlled in dependence of the water level at Schönebeck gauge (observation point 1). In the coupled simulation the water level at the observation point "Magdeburg" (observation point 2) remains below the flood warning level of 54.8, because the weir has been opened after the water level at Schönebeck gauge has reached 57. At the bifurcation point the water divided into the Old Elbe branch which results in a lower water level in the main branch of the Elbe.

Acknowledgement and Disclaimer This chapter was derived from the FP7 project CIPRNet, which has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 312450.

The contents of this chapter do not necessarily reflect the official opinion of the European Union. Responsibility for the information and views expressed herein lies entirely with the author(s).

References

1. Konikow LF, Bredehoeft JD (1992) Ground-water models cannot be validated. Adv Water Resour 15:75–83
2. Morita M, Yen BC (2000) Numerical methods for conjunctive two-dimensional surface and three-dimensional sub-surface flows. Int J Numer Meth Fluids 32:921–957
3. Morita M, Yen B (2002) Modeling of conjunctive two-dimensional surface-three-dimensional subsurface flows. J Hydraul Eng 128(2):184–200. doi:10.1061/(ASCE)0733-9429(2002)128:2(184)
4. Fairbanks J, Panday S, Huyakorn PS (2001) Comparisons of linked and fully coupled approaches to simulating conjunctive surface/subsurface flow and their interactions. In: MODFLOW 2001 and other modeling odysseys—conference proceedings, Golden, CO, p 356–361
5. Huang G, Yeh GT (2006) An integrated media, integrated processes watershed model—wash123d: part 3—a comparative study on different surface water/groundwater coupling approaches. In: Binning PJ, Engesgaard PK, Dahle HK, Pinder GF, Gray WG (eds) Proceedings of the XVI international conference on computational methods in water resources, Copenhagen, Denmark.
6. Becker B, Talsma J (2013) On the external and iterative coupling of multiple open channel flow models with OpenMI. Rev Ing Innova 6:55–66
7. Becker BPJ, Schüttrumpf H (2010) An OpenMI module for the groundwater flow simulation programme Feflow. J Hydroinform 13(1):1–13. doi:10.2166/hydro.2010.039. URL http://www.iwaponline.com/jh/013/jh0130001.htm
8. Werner M, Schellekens J, Gijsbers P, van Dijk M, van den Akker O, Heynert K (2013) The delft-FEWS flow forecasting system. Environ Model Softw 40:65–77. doi:10.1016/j.envsoft.2012.07.010. URL http://linkinghub.elsevier.com/retrieve/pii/S1364815212002083
9. Moore R, Gijsbers P, Fortune D, Gregersen J, Blind M (2005) OpenMI document series: part A—scope for the OpenMI, version 1.0 edn. URL http://www.openmi.org/openminew/documents/A_OpenMI_Scope.pdf
10. Gregersen J, Gijsbers P, Westen S (2007) OpenMI: Open modelling interface. J Hydroinformatics 9(3):175–191. doi:10.2166/hydro.2007.023
11. Moore RV, Tindall CI (2005) An overview of the open modelling interface and environment (the OpenMI). Environ Sci Policy 8(3):279–286. doi:10.1016/j.envsci.2005.03.009

12. Bulatewicz T, Yang X, Peterson JM, Staggenborg S, Welch SM, Steward DR (2010) Accessible integration of agriculture, groundwater, and economic models using the open modeling interface (OpenMI): methodology and initial results. Hydrol Earth Syst Sci 14 (3):521–534. doi:10.5194/hess-14-521-2010. URL http://www.hydrol-earth-syst-sci.net/14/521/2010/
13. OpenMI Association (2014) OpenMI. URL http://www.openmi.org
14. Gijsbers P, Gregersen J, Westen S, Dirksen F, Gavardinas C, Blind M (2005) OpenMI document series: Part B—Guidelines for the OpenMI. IT Frameworks (HarmonIT), version 1.0 edn. URL http://www.openmi.org/openminew/documents/B_Guidelines.pdf
15. Becker BPJ, Forberig S, Flögel R, Schüttrumpf H, Köngeter J (2011) On the determination of groundwater levels for hazard maps of groundwater head rise induced by high water. Wasserwirtschaft 12:10–16 (in German)
16. Becker B, Dahm R, van Heeringen KJ, Goorden N, Kramer N, Kooij K, Gooijer J, Jansen J (2012a) Op zoek naar een optimaal ontwerp voor een groot uitwateringsgemaal in het Lauwersmeer. H_2O 44(13):11–13 (in Dutch)
17. Becker B, Gao Q (2012) Koppelen Sobek-modellen "Wetterskip Fryslân" en "Waterschap Noorderzijlvest" via OpenMI. Report 1204514-000-ZWS-0007, Deltares, Delft (in Dutch)
18. Becker BPJ, Talsma J, Gao Q, Ruijgh E (2012c) Coupling of multiple channel flow models with OpenMI. In: Proceedings of 10th international conference on hydroinformatics, Hamburg, Germany
19. Schellekens J, Becker BPJ, Donchyts G, Goorden N, Hoogewoud JC, Patzke S, Schwanenberg D (2012) OpenStreams: open source components as building blocks for integrated hydrological models. In: Geophysical research abstracts, Vienna, Austria, vol 14 EGU2012, p 3953. URL http://adsabs.harvard.edu/abs/2012EGUGA..14.3953S
20. Becker BPJ, Schwanenberg D, Schruff T, Hatz M (2012b) Conjunctive real-time control and hydrodynamic modelling in application to Rhine River. In: Proceedings of 10th international conference on hydroinformatics, TuTech Verlag TuTech Innovation GmbH, Hamburg, Germany
21. Becker B, Schruff T, Schwanenberg D (2014) Modellierung von reaktiver Steuerung und Model Predictive Control (Modelling of reactive control and model predictive control). In: Selbstverlag der Technischen Universität Dresden (Simulation techniques and models for hydraulic engineering and water management), Dresden, Wasserbauliche Mitteilungen, vol 50, p 165–174 (in German)
22. Schwanenberg D, Becker BPJ, Xu M (2015) The open RTC-Tools software framework for modeling real-time control in water resources systems. J Hydroinform 17(1):130–148. doi:10.2166/hydro.2014.046
23. Becker B (2013) Inzet RTC-Tools voor het boezemmodel "wetterskip fryslân". Report 1205773-000, Deltares, Delft (in Dutch)
24. Stelling GS, Duinmeijer SPA (2003) A staggered conservative scheme for every froude number in rapidly varied shallow water flows. Int J Numer Methods Fluids 43:1329–1354
25. Deltares (2014) SOBEK 3/ D-Flow 1D and D-Real time control in delta shell/User manual. Deltares, Delft, version: 3.2.1.31793

26. Deltares (2013) RTC-Tools a toolbox for real-time control of hydraulic structures. URL http://oss.deltares.nl/web/rtc-tools, published: Deltares
27. Deltares (2012) RTC-Tools a software package for modelling real-time control/Technical reference manual and configuration guidelines. Deltares, Delft, version: 0.1.0.22313
28. Becker BPJ (2010) Zur gekoppelten numerischen Modellierung von unterirdischem Hochwasser. Dissertation, RWTH Aachen, Fakutlät für Bauingenieurwesen, Aachen. URL http://d-nb.info/100850209X/34 (in German)