An OpenMI module for the groundwater flow simulation programme Feflow

Bernhard P. Becker and Holger Schüttrumpf

ABSTRACT

The OpenMI standard defines an interface that allows time-dependent models to exchange data at runtime. The migration of a flow simulation programme to OpenMI compliance usually requires changes in the source code or even a reorganisation of the programme sequence. Users of commercial flow simulation programmes depend on the software producer if they want to couple their models according to the OpenMI standard. We describe how we made the groundwater flow simulation programme Feflow OpenMI compliant without changing the source code: an OpenMI-compliant control application communicates with Feflow via remote procedure calls. A basic prerequisite for applying this method is an entry point into the flow simulation programme that allows to modify the model data during runtime and to implement the remote procedure calls. Feflow meets this requirement as it provides the interface manager (IFM). The mode of operation is explained with a simple test case including an inundation model and a Feflow groundwater model.

Key words | coupled surface—subsurface flow, Feflow, Ilmoflood, OpenMI, remote procedure call, subsurface flood

Bernhard P. Becker (corresponding author)
Holger Schüttrumpf
Lehrstuhl und Institut für Wasserbau und Wasserwirtschaft,
RWTH Aachen University,
Mies-van-der-Rohe-Straße 1,
52056 Aachen,
Germany
Tel.: +49 241 80-25263
Fax: +49 241 80-22348
E-mail: *Bernhard.Becker@rwth-aachen.de*

INTRODUCTION

The OpenMI standard defines an interface that allows time-dependent models to exchange data at runtime (Moore *et al.* 2005). Model components that comply with the OpenMI standard can, without any programming, be coupled to OpenMI modelling systems (Moore & Tindall 2005; Gregersen *et al.* 2007). 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.

Feflow (Trefry & Muffels 2007; DHI-Wasy 2009*a*) is a finite element groundwater flow simulation programme which is widely used in Germany, but also has a large international user community. It is a commercial programme and has not been equipped with an OpenMI interface yet by the software producer. So it is left to the user who wants to use Feflow models in OpenMI systems to migrate Feflow to OpenMI compliance. The user can not change the source code, but Feflow provides a programming interface (the interface manager, IFM) where he can implement custom code. In this article we describe how we achieved OpenMI compliance for Feflow without changing the source code. The crucial point is the external control of a Feflow simulation run by an outside entity. Our solution to this problem is to implement remote procedure calls (RPC) into the IFM.

The first two sections of this paper make the reader familiar with the OpenMI standard and the Feflow IFM. After explaining in detail how we obtained the OpenMI compliance, we present a simple OpenMI system with a Feflow component modelling dike seepage under transient water level boundary conditions. With the help of this

doi: 10.2166/hydro.2010.039

example, we explain how the OpenMI linking mechanism works. The article closes with a discussion on the method used to achieve Feflow's OpenMI compliance and an outlook on future applications.

THE OPENMI STANDARD

The OpenMI standard is a software component interface definition for the computational core of hydrological and hydraulic models. An OpenMI-compliant model satisfies the following criteria (Gijsbers *et al.* 2005):

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.

Beside the standard interface specification, the OpenMI-association 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 (Moore *et al.* 2005). The easiest way to make a generic model OpenMI-compliant is to embed the code into a wrapper class provided by the OpenMI environment (Gijsbers *et al.* 2005). 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 to 8 from the list above.

The OpenMI environment also provides the OpenMI configuration editor. This programme supports the data exchange between different OpenMI compliant components. An OpenMI compliant model component is loaded into the OpenMI configuration editor as dynamic link library (DLL).

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.

In Figure 1, different layouts of pull-driven request-reply connections are shown. For the unidirectional chain, component A requests B for data. In order to respond 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

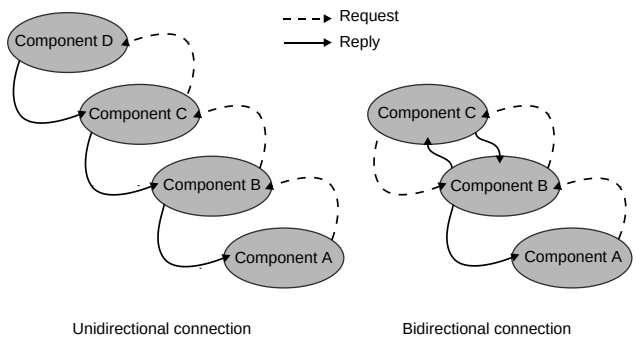

Figure 1 | Different connection layouts with the request-reply mechanism (after Gijsbers *et al.* 2005).

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 respond 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.* (2007) 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 Figure 1 (right side), components B and C would adjust their reply values iteratively as long as an accuracy criterion is fulfilled.

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.

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:

- Initialize(…)
- PerformTimeStep(…)
- Finish(…)
- 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.

Examples for OpenMI methods needed for the data exchange itself are given in the following list:

- GetCurrentTime(…)
- GetValues(…)
- SetValues(…).

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.

THE FEFLOW INTERFACE MANAGER

The Feflow interface manager is an interface which allows the user to implement custom code into Feflow at specific points within the programme sequence. The Feflow executable loads an interface manager module (IFM module) as a dynamic link library (DLL). The DLL contains so-called callback functions. The user fills these functions with his own code. The callback functions form the entry points of the DLL and are executed during the programme sequence by Feflow. Figure 2 shows the callback functions available for groundwater flow simulations. For example, the callback OnBeginSimulation is executed once before the time loop starts. PostTimeStep is invoked after the flow equation system for one time step is solved (FlowSimulation) and is invoked as many times as there are time steps. PreFlowSimulation and PostFlowSimulation enframe the FlowSimulation and hence can be invoked several times within one time step, if there are iterations to be carried out, e.g. in the case of constrained boundary conditions.

To modify model data and to retrieve model results, so-called interface manager functions (IFM functions) are available. In Table 1, examples for interface manager functions are given. IFM functions are used to set boundary conditions or to modify flow parameters and to retrieve model results during the simulation run.

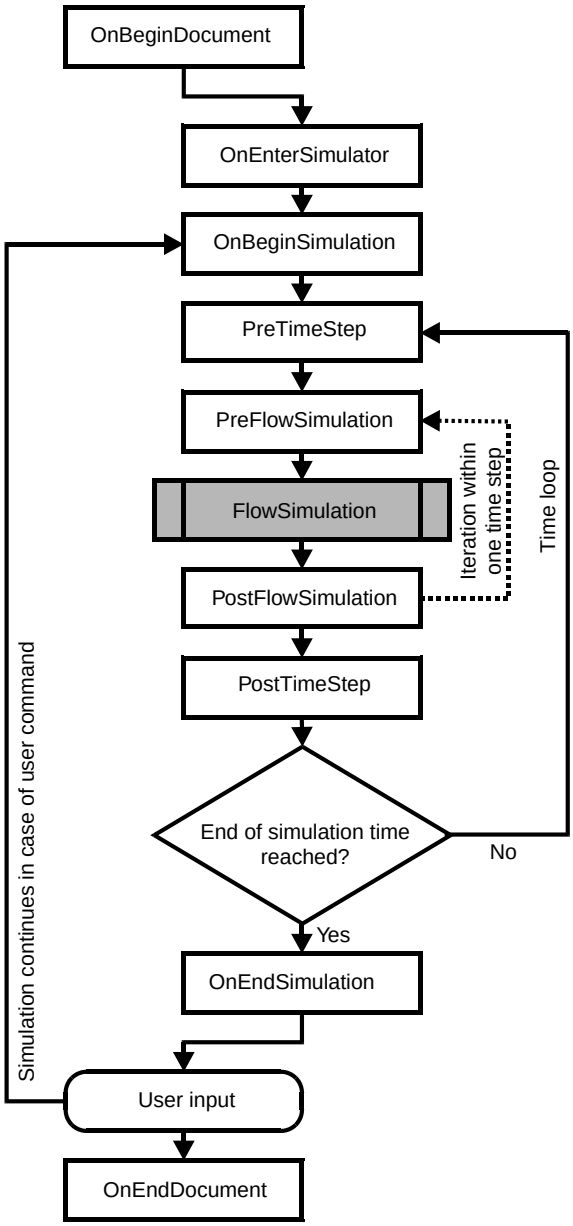

Figure 2 | Callback functions and location of the FlowSimulation, where the finite-element equation system is solved, within the programme sequence of Feflow (flow simulation only).

FEFLOW OPENMI COMPLIANCE

Because the Feflow source code is not available for this study, the code can neither be changed, reorganized, nor compiled as a DLL with the OpenMI standard functions. Thus, the runtime control by an external entity (item 1 from the list above) becomes crucial. For items 3 to 8, the OpenMI environment provides convenient features and functions. The organisation of the programme sequence (item 2) matches to the requirements, so with the beginning of the computation (in Feflow terms: simulation), the mesh geometry and boundary conditions are known.

Our strategy to reach Feflow's OpenMI compliance though is to develop an OpenMI compliant control programme that communicates with a Feflow IFM module via remote procedure calls (RPC). With remote procedure calls, one programme can execute functions from another programme and receive the return values of the function call. RPCs are based on a client-server model. One programme takes the role of a server and waits for commands from another programme, the client. Both programmes can run on one computer or on different computers.

Figure 3 shows the functional principle of the communication between the OpenMI compliant control programme and Feflow. The OpenMI configuration editor (OpenMI.exe) is shown on the left side of the figure. It loads the OpenMI compliant control programme, the FEFLOWcontroller. Independently from the Open MI configuration editor, Feflow (feflow.exe) loads the IFM module FEopenMI as shown in the sketch on the right side.

The FEFLOWcontroller and the IFM module FEopenMI communicate via synchronous RPC during runtime. The RPCs are implemented as Microsoft RPC (MSDN 2009*b*) with an interface definition according to the Microsoft Interface Definition Language (MIDL) using the named-pipe network protocol. The MIDL interface definition has been chosen, because the Feflow IFM module must be written in unmanaged C/C++ code, which is executed directly by the central processing unit (CPU). The MIDL is also based on unmanaged code and thus can be implemented directly into the Feflow IFM module. However, the OpenMI runs as C# managed code under the .NET framework and is managed by a virtual machine and not executed by the CPU. Thus, the MIDL RPC functions are coded in a separate DLL (not shown in Figure 3) as unmanaged code. This DLL is embedded into the managed OpenMI wrapper class via Platform-Invoke (MSDN 2009*a*).

Table 1 | Examples for interface manager functions of Feflow (DHI-Wasy 2009*b*)

Function name	Input value	Description
IfmGetAbsoluteSimulationTime	–	Returns the current simulation time
IfmGetResultsFlowHeadValue	Node number	Returns the computed head value at current simulation time
IfmSetBcFlowTypeAndValueAtCurrentTime	Node number, boundary condition value, boundary condition type	Sets a time constant boundary condition for the current time step and all following time steps

For the communication with the FEFLOWcontroller the callback functions

- OnBeginSimulation
- PreFlowSimulation and
- PostTimeStep

are relevant and shown in the figure. Within the initialization phase servers are set up both by the FEFLOWcontroller (server 1) and the FEopenMI (server 2). For server 1 the FEopenMI assembly acts as a client, while for server 2 the client is the FEFLOWcontroller. The server set up for the FEFLOWcontroller is located in the OpenMI initialize method, while Feflow sets up its server within the callback function OnBeginSimulation. It is necessary to use two servers, because the OpenMI method PerformTimeStep must let Feflow execute one single time step. For this task no IFM function is available.

To explain how the communication between the two DLLs FEFLOWcontroller and FEopenMI works, we start at the command server2Listen, which is located within the callback PreFlowSimulation in Figure 3. This command turns Server 2 into listening mode and so interrupts Feflow's internal time loop. In the server listening mode Feflow holds off on remote procedure calls. Feflow model data can now be requested or modified by other OpenMI components. The FEFLOWcontroller acts as a client and

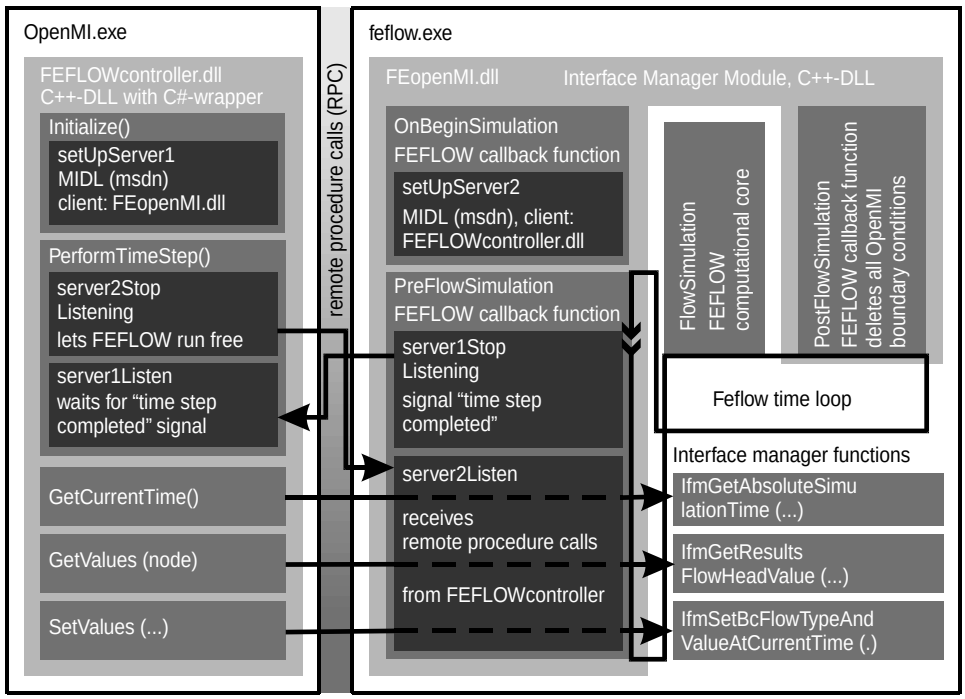

Figure 3 | Scheme of the external control of Feflow via RPC and an IFM module according to the OpenMI standard (after Becker *et al.* 2008).

forwards all requests to the FEopenMI interface manager module. For example, the call of the OpenMI method GetCurrentTime invokes the Feflow IFM function IFMgetAbsoluteSimulationTime. This function returns Feflow's current simulation time. The FEFLOWcontroller retrieves this simulation time as return value of the remote procedure call and passes it over to the OpenMI configuration editor as return value of the GetCurrentTime method. In the same manner, requests on Simulation results are obtained by the OpenMI method GetValues which has a node number as argument. The request is forwarded as two remote procedure call of the IFM function IfmGetResultsFlowHeadValue. This IFM function returns the desired head value for a specific node. Because we are in the callback PreFlowSimulation and hence the flow equation system has not been solved for the current time step yet, the return value is related to the previous time step. The OpenMI method SetValues passes its argument values to the IFM function IfmSetBcFlowTypeAndValueAtCurrentTime. This function sets a value as boundary condition at a specific node for the current time step. With the SetValues method the flow equation system for the current time step is changed according to a coupling to another model.

As mentioned above, the OpenMI standard method PerformTimeStep can not be related to an adjacent Feflow IFM function. The execution of a single time step is implemented by ending the interruption of Feflow's internal time step loop with the command server2StopListening. This command stops the listening mode of server 2. Feflow runs now free from any control of an outside entity. The next step in Feflow's internal programme sequence is to solve the equation system (FlowSimulation). Meanwhile the FEFLOWcontroller has turned into listening mode itself, because the next command within the PerformTimeStep method is server1Listen. The FEFLOWcontroller waits now for a remote procedure call from Feflow which indicates that Feflow has completed its time step. This signal is given after Feflow has passed the callback PostFlowSimulation and reaches the PreFlowSimulation callback again. Feflow acts now as client and executes the command Server1StopListening. This command ends the listening mode of the FEFLOWcontroller and allows it to complete the PerformTimeStep method. The next command Feflow executes is server2listen. Feflow is now ready for remote procedure calls from the FEFLOWcontroller again.

The cleaning of all boundary conditions within the PostFlowSimulation callback (Figure 3) after the solution of the flow equation system is necessary because all boundary conditions set with the IfmSetBCFlowTypeAndValueAtCurrentTime function are interpreted as time constant. They would remain active for the rest of the simulation if they were not deleted.

Feflow exchange items are based on node numbers (Table 2). As input exchange items, a water level, e.g. a river stage, can be set as head boundary condition (1st order boundary condition) or transfer boundary condition (3rd order, leakage boundary condition). A flow quantity can be set as well boundary condition (source or sink). Other OpenMI linkable components can request node-based water levels from a Feflow model component or a flow values related to an adjacent boundary condition.

TEST CASE "DIKE SEEPAGE"

An example for an OpenMI system including a Feflow model component is given with a dike seepage test case, where a transient water level from a surface water model component applies as boundary condition. This example has been chosen because the interchange processes are

Table 2 | Exchange items for Feflow

Type	Quantity	Dimension	ElementSet	Programme function
Input	Head	m	Node number	Head boundary condition (1st order)
Input	Head	m	Node number	Transfer (leakage) boundary condition (3rd order)
Input	Flow	m^3/d	Node number	Well boundary condition (source/sink)
Output	Head	m	Node number	Primary model result (groundwater head)
Output	Flow	m^3/d	Node number	Budget result (flow corresponding to a boundary condition)

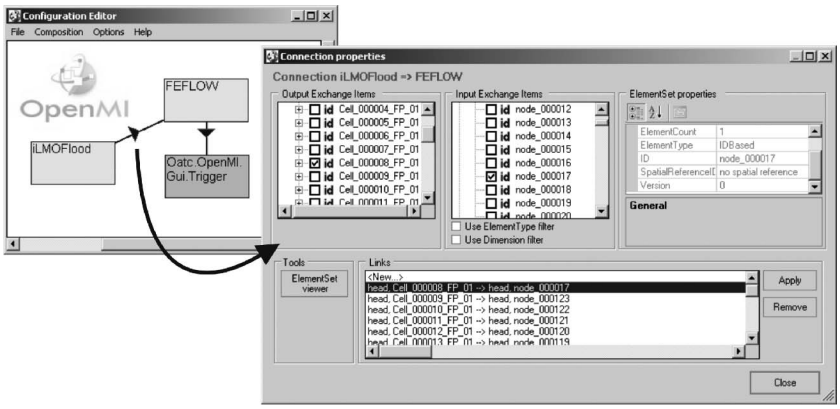

Figure 4 | Screenshot of the OpenMI editor with the OpenMI system "dike seepage" and the connection property window. The Ilmoflood and the Feflow model component are connected with links between Ilmoflood head output exchange items based on cells and the Feflow head input exchange items based on mesh nodes.

simple and easy to understand here. The dike cross-section is modelled with a two-dimensional vertical model (Figure 5), for which Feflow solves the Richards equation of variable saturated flow in porous media (see Diersch (2005) for details). At the land side slope, a seepage face boundary condition is set. The seepage face boundary condition is a head boundary condition where the head value h is the node elevation z and the flow is constrained negative to ensure that only outflow occurs:

$$h = z, \quad Q < 0 \tag{1}$$

Q is defined negative here for a flow out of the modelling area. The material parameters represent initially dry sand. The moisture content-dependent properties are modelled with a van Genuchten approach as described in WASY GmbH (2005). The time step length is 1,800 s.

The surface water flow on the foreland is modelled with Ilmoflood, a storage cell volume balance programme described first by Niemeyer & Kamrath (2007). The foreland is modelled as a chain of quadratic cells following the dike slope. At the left side of the cell chain, a transient flow boundary condition is set which increases and decreases the waterlevel in the foreland. The time step length of the Ilmoflood model is 3,600 s.

The OpenMI system is shown in Figure 4 as a screenshot of the OpenMI configuration editor. The two model components Feflow and Ilmoflood are unidirectionally connected. Feflow asks Ilmoflood for data and the data transfer direction is thus from Ilmoflood to Feflow. A back coupling as bidirectional connection, where the Ilmoflood model obtains data from Feflow as well is not intended in this test case.

The connection comprises 22 links which are listed in the connection properties window. Each link is defined between an output exchange item related to an Ilmoflood cell and an input exchange item related to a Feflow node. In Figure 5, two such links are marked as a pair of a cell number and a node number. By request of the Feflow model, the Ilmoflood model provides the OpenMI system with a head value if the cell is not dry. If necessary, this value is modified by interpolation and then passed to the Feflow model. Feflow sets a head boundary condition at the adjacent node with the retrieved value. The interpolation routine is part of the OpenMI environment.

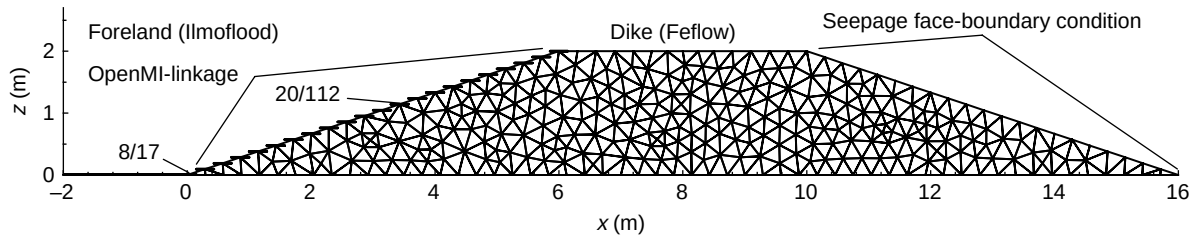

Figure 5 | Mesh of the dike cross section and the foreland, boundary conditions at the land side of the dike and the location of OpenMI links.

The trigger element initiates the computation by requesting Feflow for an arbitrary value related to the end of the simulation time. Hence, Feflow needs to run through all its time steps in order to reply on the trigger's request. But to compute one time step, the Feflow model needs head values from the Ilmoflood model. Thus, Ilmoflood computes time steps in order to reply on this request as well.

Results of the test case simulation run are shown in Figure 6. The seepage line is given as an isoline for a matrix potential $\psi = 0$ based on the pressure head values computed by Feflow. As long as the water table rises, the connection point of the seepage line with the dike slope follows the water level. At decreasing water table, the seepage line follows the water table with delay because of the flow resistance from the porous media. In the dike, water moves still towards the land side, even if the water table decreases in the foreland.

Figure 7 explains in detail the request-reply mechanism and the interpolation carried out by the OpenMI environment. To make it easier to follow the explanation, all points in simulation time are given in seconds, although the OpenMI environment compares all simulation times as modified julian days (Gijsbers *et al.* 2005). Let the Ilmoflood model have completed the computation for the simulation time point of 7,200 s. Feflow has reached a simulation time of 9,000 s. While current time steps of Ilmoflood are interpreted as completed, the current time step of Feflow is still in planning stage and will be computed next. To complete time step 9,000 s the Feflow model needs head data for all the nodes with an OpenMI link. It requests

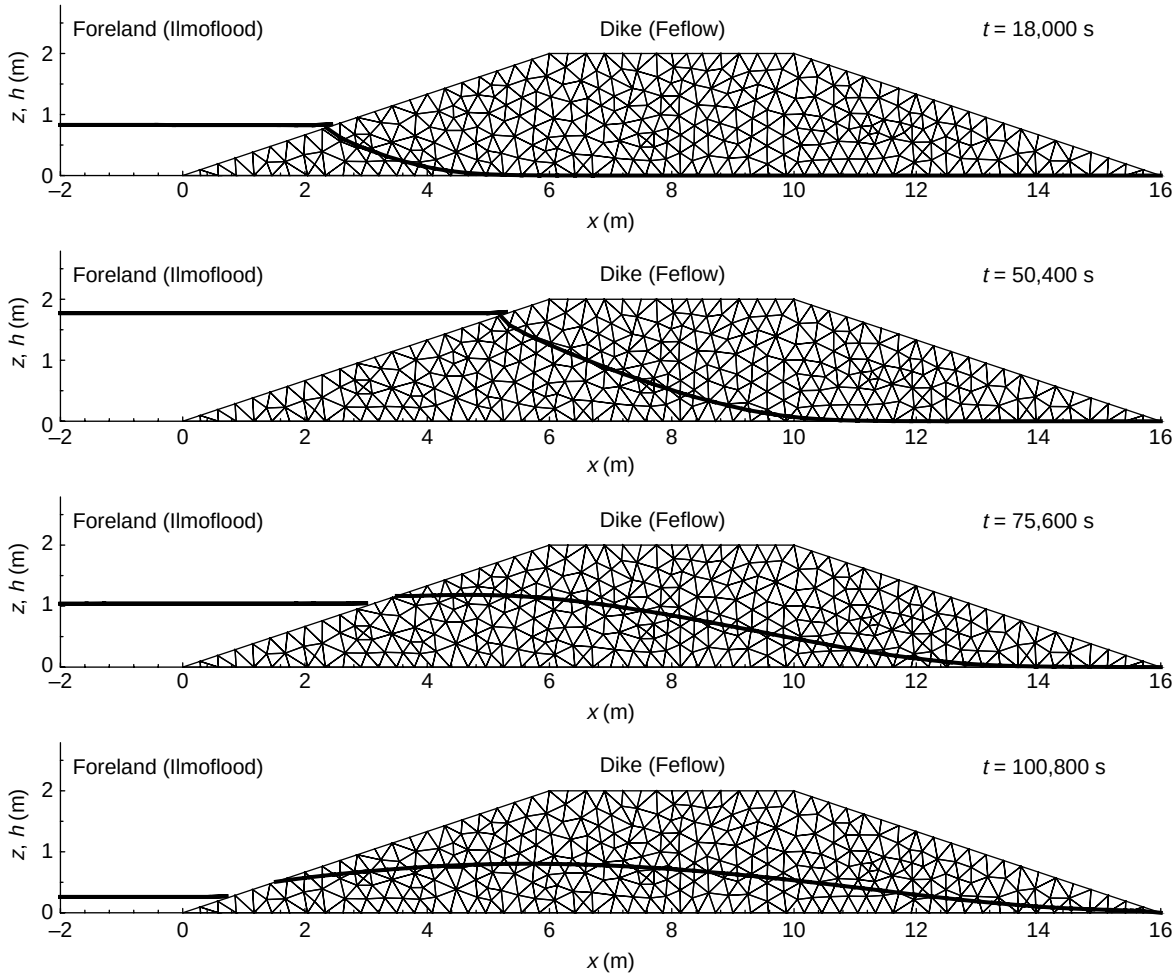

Figure 6 | Simulation results of the OpenMI system "dike seepage": seepage line and foreland water level for different time steps *t*.

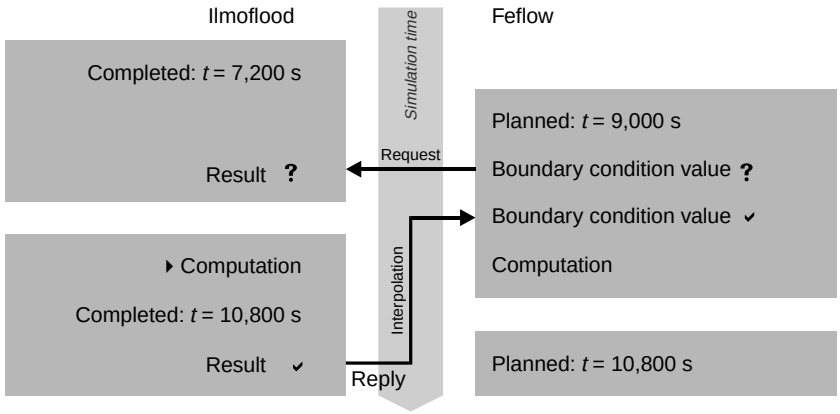

Figure 7 | Procedure of the request-reply mechanism for the uni-dirctional connection between Ilmoflood and Feflow.

this data from the Ilmoflood model. For $t = 9{,}000\,\mathrm{s}$ Ilmoflood can not provide any data at the moment, thus, the OpenMI environment lets it compute the next time step by executing the method PerformTimeStep. With $t = 10{,}800\,\mathrm{s}$ Ilmoflood's next time step is temporarily behind the requested one. The OpenMI environment interpolates values for $t = 9{,}000\,\mathrm{s}$ from $t = 10{,}800\,\mathrm{s}$ and $t = 7{,}200\,\mathrm{s}$. Values for these two time steps are stored in an OpenMI internal buffer for all Ilmoflood cells linked to another model component. With the interpolated values the Feflow model is able to compute the planned time step. For Feflow's next time step ($t = 10{,}800\,\mathrm{s}$) the Ilmoflood model component can reply without any further calculations.

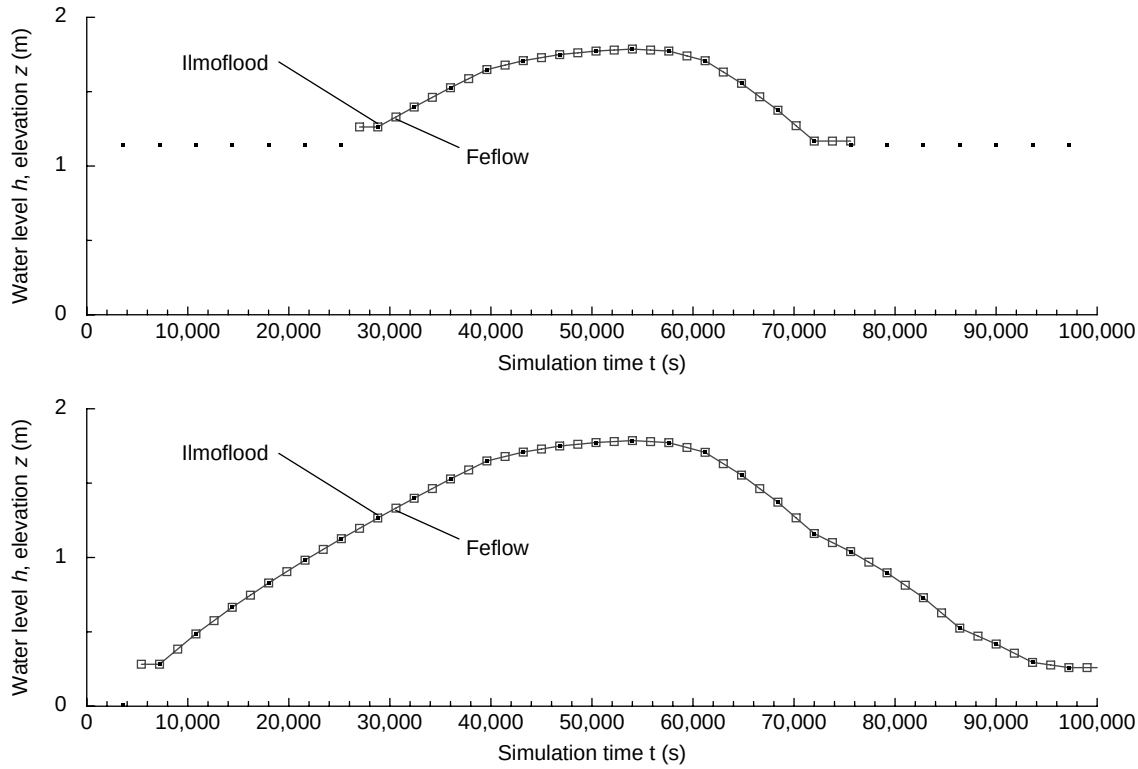

Figure 8 | Results of the Ilmoflood model (water level or elevation in case of dry cell) and adjacent boundary conditions set in the Feflow model for the links between cell 20 and node 112 (top) and cell 8 and node 17 (bottom).

The hydrographs of the Ilmoflood model results and the boundary condition set in the Feflow model are compared in Figure 8 for two different cell-node links. Because the Feflow model works on a finer time grid than the Ilmoflood model, Feflow receives interpolated values for every second time step as described above. For the link between cell 20 and node 112 (Figure 8, top) there are time spans at the beginning and the end of the simulation where no boundary conditions are set in the Feflow model because the cells are dry in the Ilmoflood model for these time spans. If only one value is available for the interpolation, this interpolation boundary is passed to the Feflow model.

DISCUSSION, CONCLUSIONS AND OUTLOOK

In this article we present how we achieved the OpenMI compliance of Feflow. An OpenMI compliant component communicates via remote procedure calls with an IFM module load by Feflow. In comparison to the migration method proposed by Gijsbers *et al.* (2005), a drawback of the RPC method presented here is that it is more laborious to code. Hence, the RPC method is mainly appropriate for users who want to couple models with the OpenMI standard, but do not have access to the source code. One prerequisite for applying this method is that the flow programme to be made OpenMI compliant has an entry point which allows to modify the model data during runtime and to implement the remote procedure calls. However, the method can also be interesting for the software producer as well, because making such a powerful programme like Feflow OpenMI-compliant is a huge task and the required reorganisation of the code will probably lead to a new programme version. With the RPC method the OpenMI compliance does not affect the main programme, because all OpenMI-related features are coded in an IFM module separately. The IFM module and the controller programme can be handed out to interested users apart from the main programme.

In the OpenMI developer forum (see The OpenMI Association 2009) individual problems concerning the OpenMI compliance of commercial programmes sometimes come up. With the described method the users are free to reach an OpenMI compliance according to their individual needs. For the test case shown in this article, for example, only exchange items based on the groundwater flow simulation are relevant. Other Feflow-OpenMI applications may focus on termohaline and solute transport or use exchange items based on points, lines, or polygons instead of node numbers. Furthermore, each OpenMI model component is responsible for the interpretation of the exchanged values. For this study, the way how Feflow interprets a point in time is designed for the implementation of head boundary conditions coming from a surface flow model. A point in time is interpreted by Feflow as to be computed, while Ilmoflood considers it as already completed. For other problems, it could be purposive to let Feflow interpret a point in time in a different way.

For the test case shown, the major benefit of the OpenMI coupling is the simplification of the boundary condition generation. Because of the slope and the moving water table, each node needs its own boundary condition hydrograph to model the movement of the waterlevel. For all nodes, the hydrographs must be well matched to form a scenario. Within the OpenMI system, the boundary conditions are obtained directly from the water level simulation carried out by Ilmoflood. The modeller does not have to carry out the interpolations and unit conversions between Feflow (time in days) and Ilmoflood (time in seconds) anymore. While for the test case shown here the generation of boundary condition hydrographs for 22 nodes would not be that laborious, for more complex cases, the use of OpenMI components is expected to be more valuable. Test runs not described here showed, that the RPC-method is also applicable for larger models. The experience we made is that the computation time of the OpenMI coupled system still depends mainly on the model components, i.e. the solution of the flow equations, while the data exchange contributes only marginally to the total computation time.

Because of the request-reply-mechanism, where one model component waits for data from another model component, the computation time of all model components adds up. Thus, within an uni- or bidirectional coupled OpenMI system the model components can basically not run parallel. However, the method described within this article allows to run Feflow and the OpenMI system on different machines, if the network protocol of the remote

procedure calls is changed to a different type. This could be useful if different operating systems shall be used.

If bidirectional links were implemented for the dike seepage test case, the water seeping into the dike would be withdrawn from the Ilmoflood model. When the Feflow model requests a value from the Ilmoflood model, the Ilmoflood model itself needs a value from the Feflow model. Feflow would provide a guess based on previous simulation results to allow Ilmoflood to complete its time step. This guess is self-evident not that accurate as it was if the exchange values were adjusted within an iterative coupling. The OpenMI standard also supports this iterative coupling, too, but for this coupling layout additional OpenMI functions are to be implemented. The extension of the code in order to enable iterative coupling is planned as future development. Therefore the model component must be able to repeat the computation of one time step. The Feflow IFM provides functions to carry out this repetition (see e.g. Becker *et al.* 2009).

It is planned to use OpenMI systems with a Feflow groundwater model and an Ilmoflood inundation model to address subsurface flood problems. A subsurface flood is a large rise of the groundwater level due to a flood event in a river. The relevant water interchange processes are bank storage (see e.g. Pinder & Sauer 1971; Ubell 1987) and the infiltration of water from an inundated area. Because of the interchange processes at least a bidirectional model coupling is necessary for predictive simulations. A severe subsurface flood event occurred in the City of Dresden, Germany, during the flood event in August 2002 (Huber *et al.* 2003; Sächsisches Landesamt für Umwelt und Geologie 2003; Walther & Marre 2004; Sommer & Ullrich 2005; Kreibich & Thieken 2008). While the City of Dresden is now well prepared with a forecasting tool based on coupled models (Sommer *et al.* 2008), for other sites there is still a demand in such forecasting tools. Having made Feflow OpenMI compliant, it is now possible to use site-specific groundwater models which already exist for the planning of measures against subsurface flood. These models usually have not been developed for the subsurface flood protection purpose, but, for example, to support water management decisions. The usage of those models for subsurface flood management within an OpenMI system has several advantages: firstly, no extra models have to be developed, secondly, the groundwater modeller's experience comes to bear on the flood protection process and thirdly, the flood protection planning benefits from continuously maintained models. That already existing (site-specific) models can be integrated in coupled modelling systems because more and more flow simulation programmes are being equipped with an OpenMI interface (see The OpenMI Association 2009), is in our opinion an important fact that will facilitate coupled model simulations in future and will bring them closer towards the state of the art.

ACKNOWLEDGEMENTS

The authors thank R. Gründler from the Feflow support team for his ideas concerning the external control of Feflow. The authors also thank the two anonymous reviewers whose comments helped to improve the manuscript.

REFERENCES

Becker, B., Reuter, C. & Schüttrumpf, H. 2008 An openMI-connected surface flow – subsurface flow simulation model system to forecast subsurface flood. *Poster presented at the Computational Methods in Water Resources XVII International Conference, 06.-10-07.2008, San Francisco, USA*. http://www.iww.rwth-aachen.de/research/papers/Becker_CMWR2008.pdf.en

Becker, B., Nowack, L., Klauder, W. S., Köngeter, J. & Schüttrumpf, H. 2009 A nonlinear leakage boundary condition for modelling flood induced groundwater head rise. *Wasserwirtschaft* **99** (1–2), 27–31.

DHI-Wasy 2009*a* *FEFLOW*. www.feflow.de, checked 2009-03-13.

DHI-Wasy 2009*b* *Feflow Online Help Feflow 5.3*. checked 2009-05-12.

Diersch, H. J. 2005 *FEFLOW Finite Element Subsurface Flow and Transport Simulation System: Reference Manual*. WASY GmbH, Berlin.

Gijsbers, P., Gregersen, J., Westen, S., Dirksen, F., Gavardinas, C. & Blind, M. 2005 *Open-MI Document Series: Part B–Guidlines for the OpenMI*. IT Frameworks (HarmonIT).

Gregersen, J. P., Gijsbers, P. J. A. & Westen, S. J. P. 2007 OpenMI: open modelling interface. *J. Hydroinform.* **9** (3), 175–191.

Huber, G., Hiller, G. & Braune, A. 2003 Konzepte des Hochwasserschutzes für die Bauten des Freistaates Sachsen im Historischen Stadtkern von Dresden. In *Umweltamt Landeshauptstadt Dresden, Dresdner Grundwasserforschungszentrum e.V (Editors), Hochwassernachsorge Grundwasser Dresden: Wissenschaftliche Tagung zum BMBF-Forschungsprojekt/ 8.Oktober 2003/Dresden, Rathaus/Tagungsband*, pp. 57–62. Druckerei und Verlag Christoph Hille, Dresden.

Kreibich, H. & Thieken, A. H. 2008 Assessment of damage caused by high groundwater inundation. *Water Resour. Res.* **44** (9), W09409.

Moore, R., Gijsbers, P., Fortune, D., Gregersen, J. & Blind, M. 2005 *OpenMI Document Series: Part A–Scope for the OpenMI*. IT Frameworks (HarmonIT).

Moore, R. V. & Tindall, C. I. 2005 An overview of the open modelling interface and environment (the OpenMI). *Environ. Sci. Policy* **8** (3), 279–286.

MSDN 2009*a Interoperating with unmanaged code*. http://msdn2.microsoft.com/en-us/library/sd10k43k.aspx, checked 2009-03-10.

MSDN 2009*b Remote Procedure Call*. http://msdn2.microsoft.com/en-us/library/aa378651.aspx, checked 2009-03-10.

Niemeyer, M. & Kamrath, P. 2007 Modellierung der Breschenbildung und Überflutung – vom Versagen zu den Folgen. In *37.IWASA, Internationales Wasserbau-Symposium Aachen 2007: Sicherheit und Risiko wasserbaulicher Anlagen. Lehrst. u. Inst. für Wasserbau u. Wasserwirtschaft RWTH Aachen, Mitteilungen* (ed. J. Köngeter), pp. I1–I29. Shaker, Aachen.

Pinder, G. F. & Sauer, S. P. 1971 Numerical simulation of flood wave modification due to bank storage effects. *Water Resour. Res.* **7** (1), 63–70.

Sächsisches Landesamt für Umwelt und Geologie 2003 *Einfluss des August Hochwassers 2002 auf das Grundwasser*. Materialien zur Wasserwirtschaft, Dresden.

Sommer, T., Ettrich, N., Eulitz, K., Haase, D., Karpf, C., Peetz, J. -V., Steckel, B. & Weichel, T. 2008 Entwicklung eines 3-Zonen-Modells für Grundwasser- und Infrastruktur-management nach extremen Hochwasserereignissen in urbanen Räumen ("3ZM-GRIMEX"). Abschlussbericht/ BMBF-Verbund-Projekt/FKZ: 02WH0557, Dresdner Grundwasserforschungszentrum e.V (DGFZ); Fraunhofer Institut für Algorithmen und Wissenschaftliches Rechnen (SCAI); Fraunhofer Institut für Techno- und Wirtschaftsmathematik (ITWM); Technische Universität Dresden, Institut für Siedlungs- und Industriewasserwirtschaft (ISI); Helmholtz-Zentrum für Umfeltforschung GmbH - UFZ, Department Angewandte Landschaftsökologie. http://edok01.tib.unihannover.de/edoks/e01fb09/594029457.pdf

Sommer, T. & Ullrich, K. 2005 Auswirkungen des Hochwassers 2002 auf das Grundwasser. Forschungsbericht/Hochwasser/ Nachsorge/Grundwasser/Dresden, http://edok01.tib.uni-hannover.de/edoks/e01fb06/507079361.pdf

The OpenMI Association 2009 *OpenMI*. www.openmi.org, checked 2009-05-12.

Trefry, M. G. & Muffels, C. 2007 Feflow: a finite-element ground water flow and transport modeling tool. *Ground Water* **45** (5), 525–528.

Ubell, K. 1987 Austauschvorgänge zwischen Fluß und Grundwasser–Teil I. *Deutsche Gewässerkundliche Mitteilungen* **31** (4), 119–125.

Walther, W. & Marre, D. 2004 Kopplung Fließgewässer–Grundwasser/Wirkung des Hochwassers auf den unterirdischen Raum. In *36.IWASA Internationales Wasserbau-Symposium Aachen 2006. Mitteilungen des Inst. f. Wasserbau und Wasserwirtschaft, RWTH Aachen*, pp. 173–202. Shaker, Aachen.

WASY GmbH 2005 *Wasy Software FEFLOW Finite Element Subsurface Flow & Transport Simulation System: White Papers, 1*. Wasy GmbH, Berlin.

First received 23 September 2009; accepted in revised form 12 October 2009. Available online 19 March 2010