



- Date:** 18<sup>th</sup> – 20<sup>th</sup> January 2005 09:00 am – 16.30 pm (closure 20<sup>th</sup> at 15.00 pm)
- Venue:** WL Delft Hydraulics, Rotterdamseweg 185, Delft
- Lecturers:** Stef Hummel WL Delft Hydraulics  
 Rob Brinkman WL Delft Hydraulics  
 Peter Gijssbers WL Delft Hydraulics
- Objective:** The objectives of the training is to learn the concepts of OpenMI, to learn how to build an OpenMI component from scratch and to learn how to migrate a legacy code into an OpenMI compliant component
- Preconditions:** The course assumes proper knowledge of Object Orientation and preferably working experience with C# or another OO-language. If this knowledge is not available, a C# course is advised before joining an OpenMI course.

## Agenda

### Day 1: 18<sup>th</sup> January 2005

9.15-10.15	Introduction to OpenMI	plenary presentation
10.15-10.30	Demo of OpenMI	plenary demo
	Coffee break	
10.45-16.30	Develop Use Case 1	individual work

Lunch will be around 12.30 h, tea breaks as requested

### Day 2: 19<sup>th</sup> January 2005

9.15-10.00	Introduction to model migration for OpenMI	plenary presentation
10.00-16.30	Develop Use Case 2	individual work

Lunch will be around 12.30 h, coffee and tea breaks as requested

### Day 3: 20<sup>th</sup> January 2005

The schedule of Day 3 completely depends on the progress made in day 1 and 2, and the discussion items raised.

9.15-10.00	Plenary discussion of items raised during migration	plenary discussion
10.00-lunch	Develop Use Case 3	individual work
lunch-14.00	Advanced issues with OpenMI	plenary presentation
14.00-14.15	Planning migration	plenary presentation
14.15-14.30	The future of OpenMI	plenary presentation
14.30-15.00	Miscellaneous issues, questions, discussions	plenary discussion
15.00	closure of the course	plenary

Lunch will be around 12.30 h, coffee and tea breaks as requested

## Use Case 1: the DataCombinator

Using C#, a Linkable Component (the DataCombinator) will be build from scratch which requests for values from two files and returns the average value.

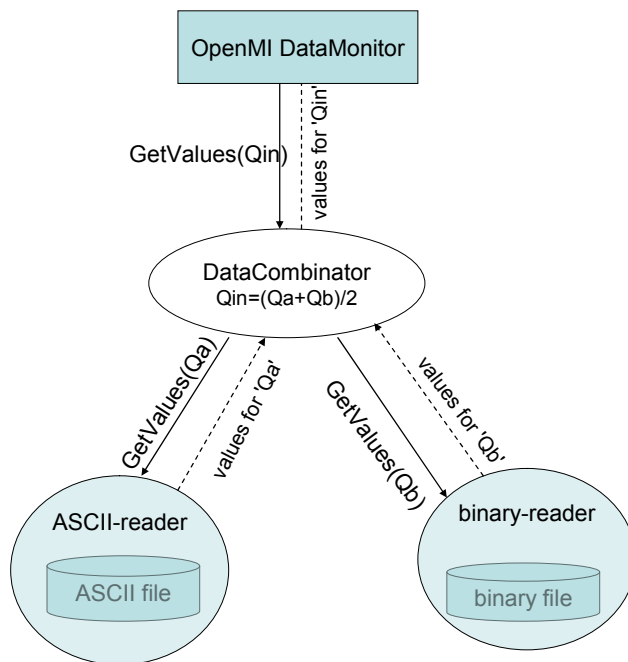


Figure 1 Use Case 1: DataCombinator averages data from a binary and an ASCII file

### Participant's role:

- model coding

### Preconditions:

The participant has:

- a running application of the VisualStudio.NET 2003 development environment on its PC
- the org.OpenMI.assemblies running on its PC (including the OpenMI DataMonitor)
- two Linkable Components on its PC:
  - the binary reader (source code, assembly/DLLS, OMI-file, binary data file)
  - the ASCII-reader (source code, assembly, OMI-file, ASCII time series file)

### Success guarantee:

- Using the OpenMI data monitor, the average value of the two data sets can be shown on screen
- Using the NUnit framework, public methods have shown to pass the test

### Main success scenario:

The participant

1. loads the OpenMI GUI on the PC and uses the GUI to browse for available Linkable Components
2. finds the OMI-files for the binary reader, the ASCII reader, the DataCombinator and the DataMonitor
3. loads the three files in the GUI and displays them on the canvas
4. creates three uni-directional, ID-based links (see figure 1), and defines the link-properties by selecting relevant quantities, element sets and (if relevant) additional data operations
5. defines the output time and runs the system

### Possible extension:

- Add more or configurable data operations to the DataCombinator

## Use Case 2: Migrating the Simple River

Using F90, C# and the OpenMI-wrapper utilities, an existing F90-executable (SimpleRiver.exe) will be migrated into a Linkable Component.

This LinkableComponent can accept Qin from a connected Linkable Component, and can provide Qout to a LinkableComponent. If no Qin is connected it retrieves input from its private data file.

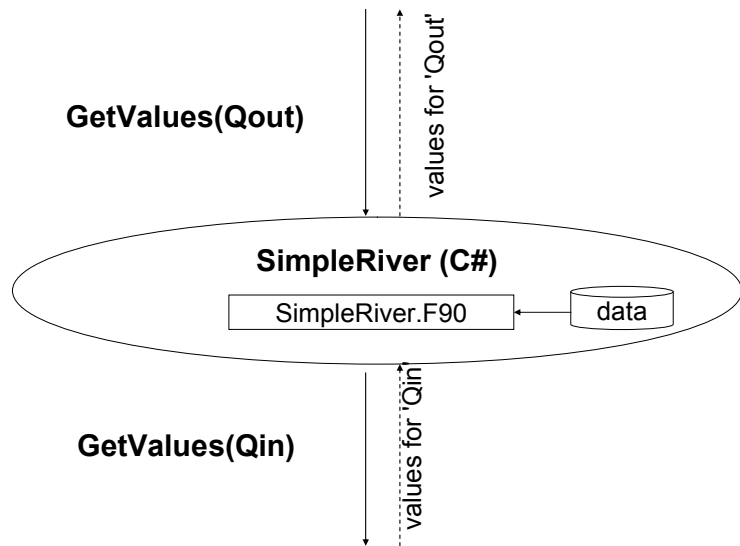


Figure 2 Use Case 2: Transfer of SimpleRiver F90-executable into OpenMI Linkable Component

### Participant's role:

- model coding

### Preconditions:

The participant has

- a running application of the VisualStudio.NET 2003 development environment on its PC
- a running application of Visual Fortran 6.6C compiler or any other F90-compiler on its PC
- the org.OpenMI.assemblies running on its PC (including the OpenMI DataMonitor)
- the org.OpenMI source code running on its PC
- three Linkable Components on its PC:
  - the DataCombinator (developed in day 1)
  - the binary reader (source code, assembly/DLLS, OMI-file, binary data file)
  - the ASCII-reader (source code, assembly, OMI-file, ASCII time series file)
- the F90-source code of the SimpleRiver.exe on its PC

### Success guarantee:

- Using the OpenMI DataMonitor, the SimpleRiver output is displayed, with the private data file as input

### Main success scenario:

The participant

1. loads the OpenMI GUI on the PC and uses the GUI to browse for available Linkable Components
2. finds the OMI-files for the SimpleRiver and the DataMonitor
3. loads the two files in the GUI and drags them to the canvas
4. creates one uni-directional, ID-based link from the DataMonitor to the SimpleRiver and defines the link properties by selecting a relevant quantity and element set
5. defines the output time and runs the system

### Use Case 3: Running the full composition

Using the OpenMI GUI, the DataCombinator and F90, C# and the OpenMI-wrapper utilities, an existing F90-executable (SimpleRiver.exe) will be migrated into a Linkable Component. This LinkableComponent can accept Qin from a connected Linkable Component, and can provide Qout to a LinkableComponent. If no Qin is connected it retrieves input from its private file.

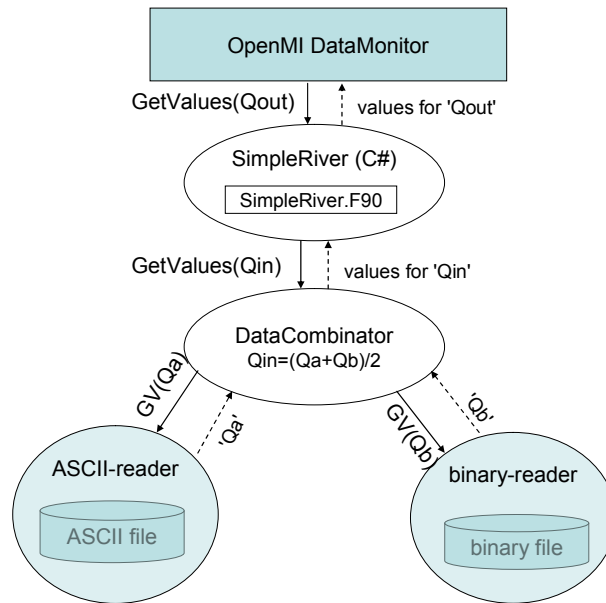


Figure 3 Use Case 3: Setting up the system

#### Participant's role:

- model integrator/builder

#### Preconditions:

The participant has:

- the org.OpenMI.assemblies running on its PC
- four Linkable Components on its PC:
  - the DataCombinator (developed in day 1, assembly, OMI-file)
  - the binary reader (assembly/DLLS, OMI-file, binary data file)
  - the ASCII-reader (assembly, OMI-file, ASCII time series file)
  - the SimpleRiver component (developed in day 2, assembly, OMI-file)

#### Success guarantee:

- Using the OpenMI DataMonitor, the SimpleRiver output is displayed, with the input data being the combination from the binary and ACSII file

#### Main success scenario:

The participant:

1. loads the OpenMI GUI on the PC and uses the GUI to browse for available Linkable Components
2. finds the the OMI-files for the SimpleRiver, the DataCombinator, the ASCII-file, the binary-file and the DataMonitor
3. loads the five files in the GUI and drags them on the canvas
4. creates five uni-directional, ID-based link (see Figure 3) and defines the link properties by selecting a relevant quantity and element set
5. defines the output time and runs the system

#### Extension (the participant's role turns into tool coder)

The participant creates its own application to run the above mentioned combination and display the results. The coder has the choice to use the Configuration package as provided by the OpenMI utilities, or it can build an application with hard coded links.