Pinios Use Case B: Impact of Climate Change Scenarios on the Reliability of a Reservoir

Migration of a Reservoir Management Model (RMM-NTUA)

Research team: A. Efstratiades, S. Kozanis, I. Liagouris, and E. Safiolea

Why linking a Reservoir Management Model with a R-R model

This case study involves linking a Reservoir Management Model with a Rainfall-Runoff model.

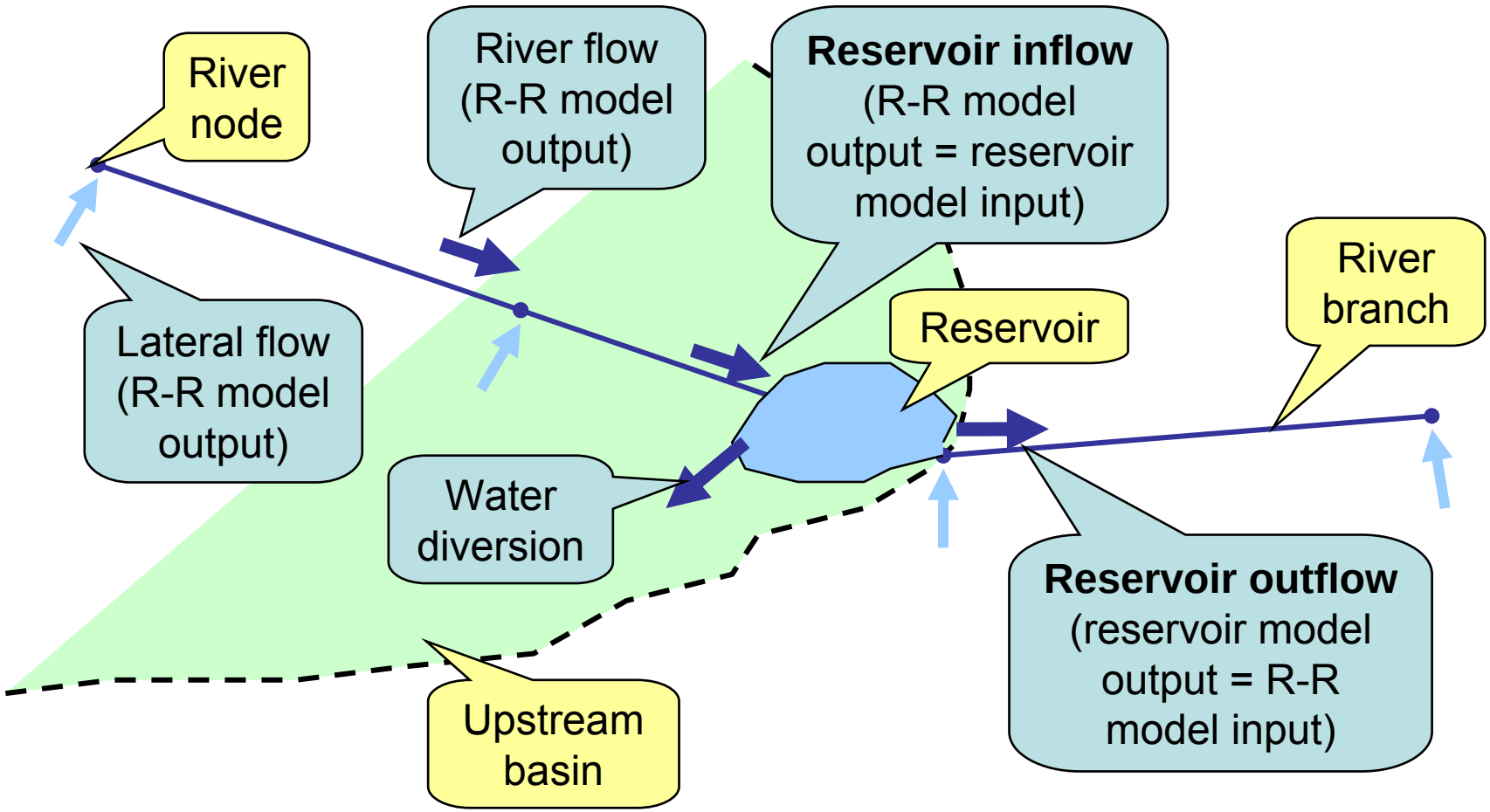
- Reservoir studies require reliable runoff data; however, the quantity and quality of historical records (if they exist) are usually inadequate.

- Hydrological models are well-established tools for predicting discharge across a river network, at various time-scales.

- Such models, especially the physically-based ones, are the only rational tools to assess the impacts of future events on runoff regime, such as land-use, vegetation, and climate changes.

- Although some hydrological models include reservoir simulation routines, the emphasis is merely given on the hydraulic processes (i.e. spillway routing) and not on the water management aspects.

Reservoir and R-R Models: Schematic Representation



Modelling Issues Related to the Case Study

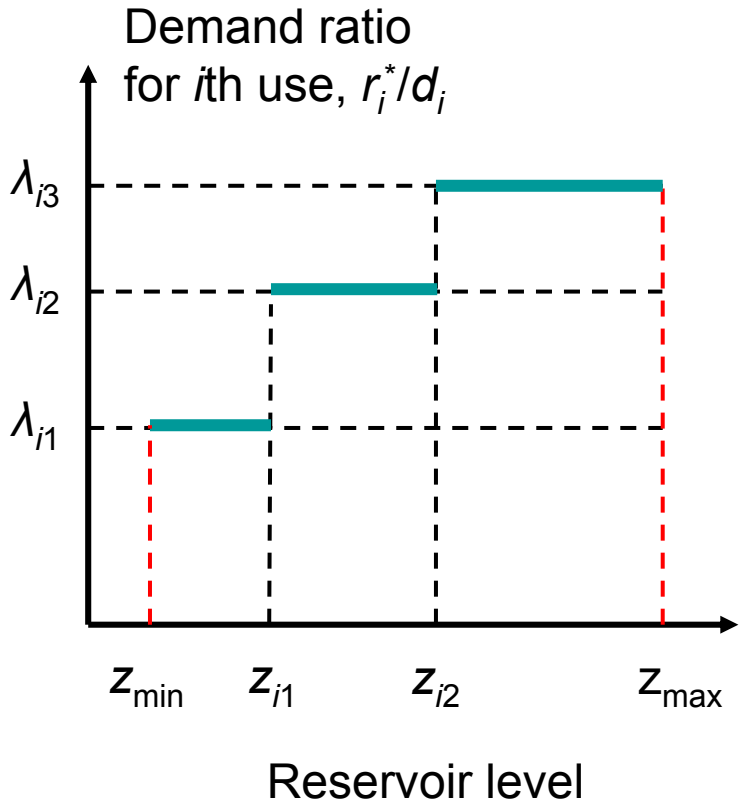
- Design a Reservoir Management Model of general purpose, compliant with the OpenMI Environment (engine written in Borland Delphi)
- Incorporate multiple water uses and assign operation rules to each use.
- Take into account all essential water balance components, including losses due to evaporation and leakage.
- Ensure flexibility regarding time-scale (from hourly to monthly).

Reservoir Simulation Model: Input Data

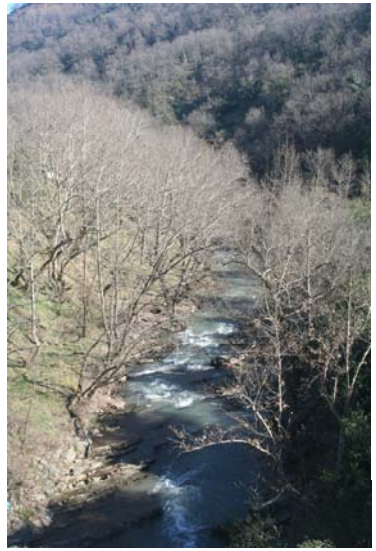
- Time step and time horizon of simulation
- Level-storage and level-surface data (given as point series)
- Characteristic levels (minimum, maximum, initial);
- Upstream watershed area;
- Time series of precipitation and evaporation depths;
- Leakage function coefficients (monthly);
- Water uses properties (priority order, demand time series, operation rules).

Operation Rules

- Operation rules specify the desirable release, *ri**, for the corresponding use *i*, as a function of the actual level, *z*.
- Desirable releases are expressed as ratios of the actual demand, *di*.
- For each use are assigned step functions, expressed as point series (λij, *zij*), where *z*min ≤ *zij* ≤ *z*max and 0 ≤ λij ≤ 1.
- There is no limit on the number of (λij, *zij*) pairs.



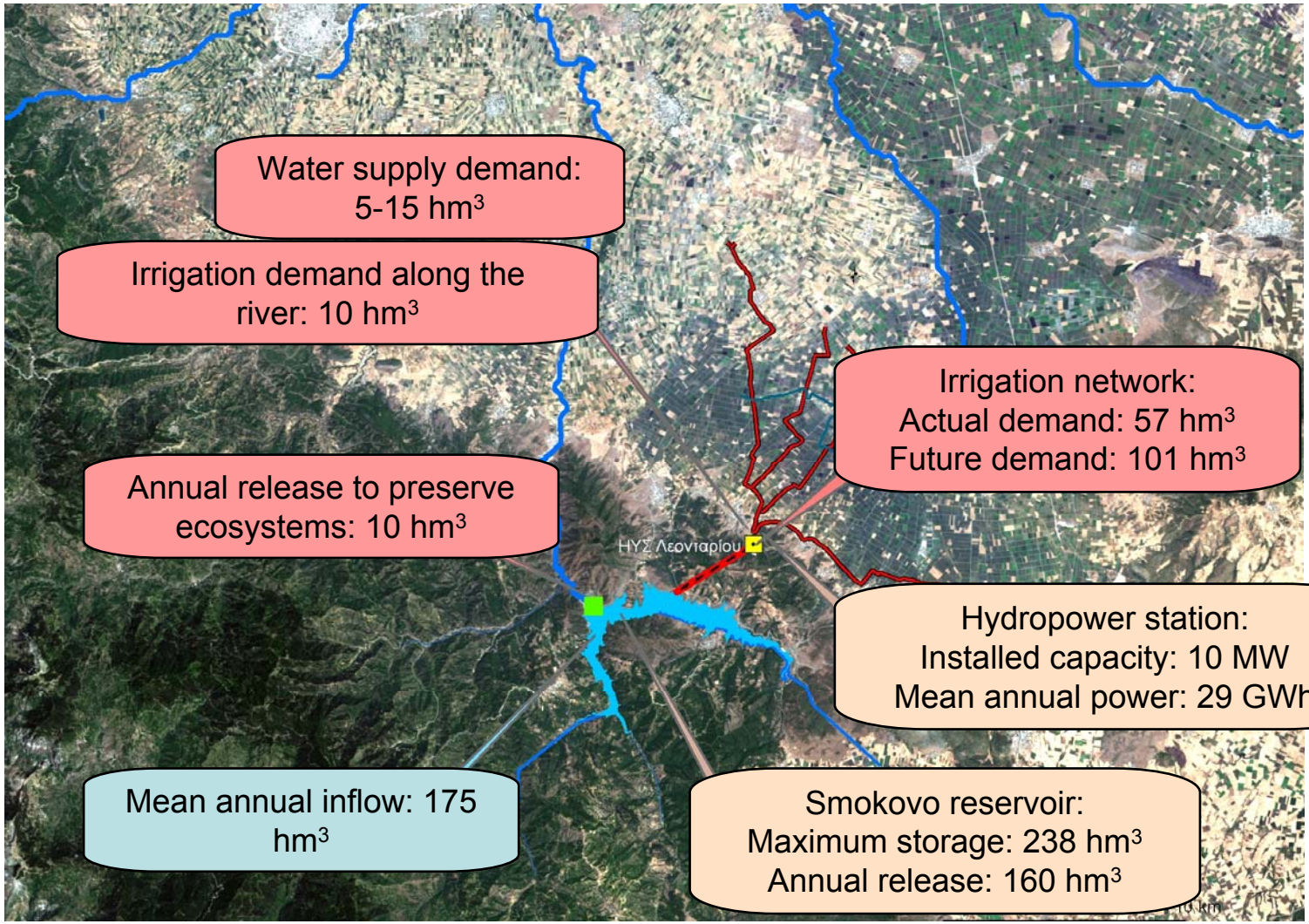
The Smokovo Reservoir



Case Study: Historical Background

- Newly constructed reservoir (pilot operation July 2002)
- Located on the SW area of Thessaly plain, a major agricultural area of Greece, suffering from severe water deficits during dry years and water table degradation, due to pumping
- Aims to supply 25 000 ha of agricultural land, through a pressured pipe network, thus limiting the extensive use of boreholes
- Today, small part of the network, 1800 ha, is finished and other 3700 ha are irrigated through small barrages
- Moreover, the reservoir will serve for providing drinking water for 55 000 residents and ensuring permanent flow downstream of the dam, during the summer period.

Case Study: The Smokovo Reservoir

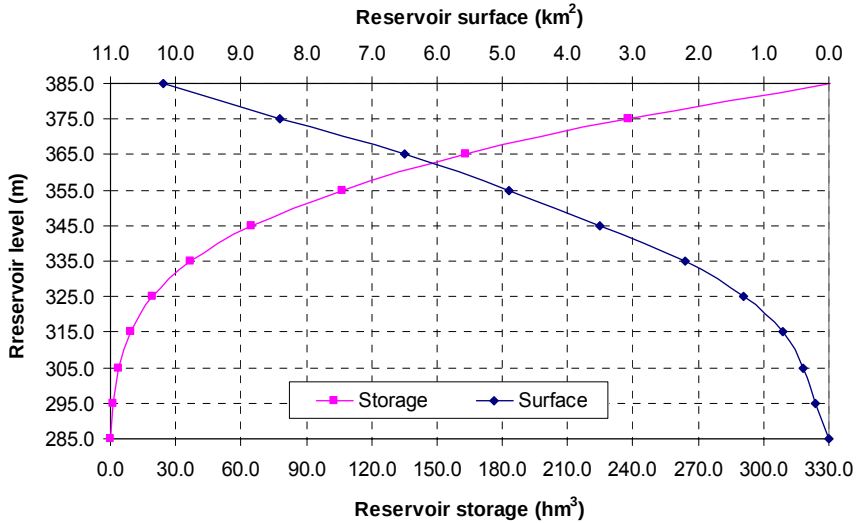


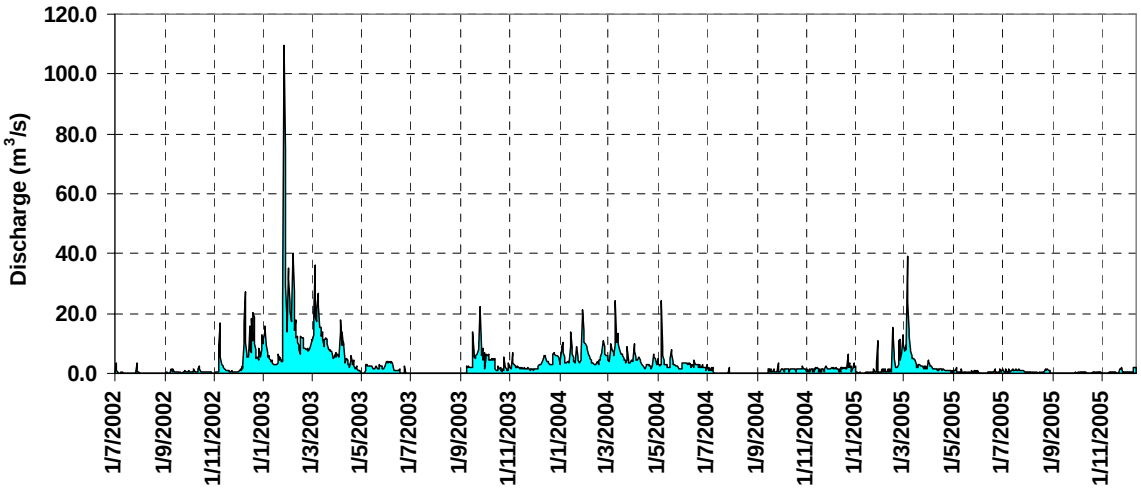
Case study: Questions to Be Answered

- Given that the irrigation network is gradually constructed, what is the optimal allocation of reservoir releases to the various uses?
- What are the impacts on downstream flow, due to various management policies?
- What are the impacts on reservoir inflows and, consequently, reservoir releases and downstream flow due to possible climatic changes?

Case study: Data for Reservoir Simulation

- Reservoir properties
- Daily rainfall and runoff series (historical, 7/2002-12/2005)
- Water demand for environmental preservation, irrigation and domestic supply (actual scenario)
- Operation rules (not optimized)





Level-storage-surface curves

Daily inflow series

Steps of Migration (1)

1. Use Cases (How the model will be used??)
2. Create a simple reservoir model (DLL)
3. Create the .NET Assemblies (Wrapper & Test Classes)
4. Create the .omi File
5. Run a simulation on OmiEd

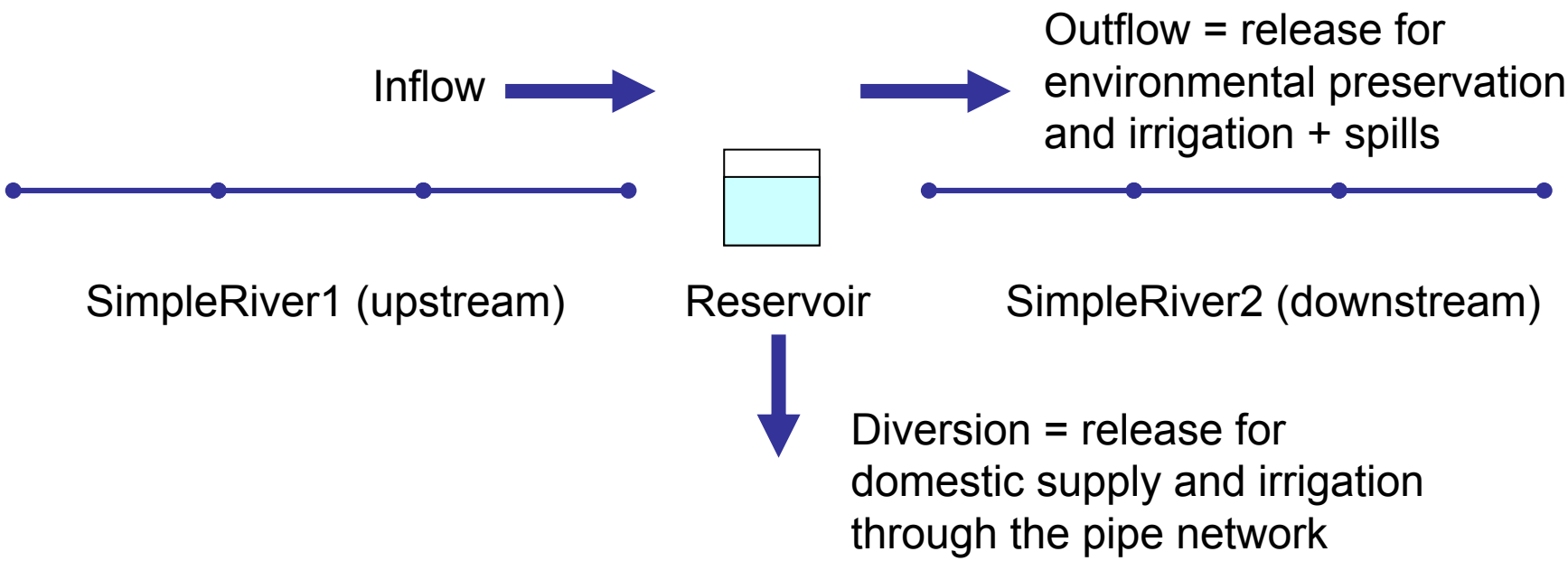
Steps of Migration (2)

Use Case :

- Reservoir is assigned to a specific river node.
- At each time step, the upstream inflow due to runoff is given by the hydrological model (remember that inflow is adjusted to take into account the actual reservoir surface, directly fed by precipitation).
- Releases are either conducted downstream or diverted; the formers are aggregated and then added to spill, to estimate total reservoir outflow.
- Outflow is returned to hydrological model.

Steps of Migration (3)

Use Case :



Steps of Migration (4)

Create a simple reservoir model (SRM) :

- One Input Exchange Item (Inflow)
- One Output Exchange Item (Outflow)
- One Input File :

  Number of Time Steps

  Time Step Length

  Initial Storage (t=0)

  Inflows (for each time step)

  Outflow (constant for all time steps)

Storage(t) = Storage(t-1) + [Inflow – Outflow]*TimeStepLength (S.I)

Steps of Migration (5)

Create a simple reservoir model (SRM):

{Engine Core before the transformation}

```
while (NumberOfTimeSteps>0) do
 begin
 Storage := Storage + (Inflow-Outflow)*TimeStepLength ;
 NumberOfTimeSteps := NumberOfTimeSteps - 1 ;
 end;
```

Steps of Migration (6)

Create a simple reservoir model (SRM) :

{Engine Core after the transformation : Building the .dll}

Functions

- **Initialize()** : For instantiation and opening input/output files **(required)**.
- **PerformTimeStep()** : For performing a single time step of the model **(required)**
- **Finish()**: For closing input/output files after a full simulation **(required)**
- **Dispose()** : For de-allocating memory after a full simulation **(required but not necessary to implement)**
- **GetModelDescription()** : For exporting a string that describes the model **(required)**
- **GetModelID()** : For exporting a string that defines the model **(required)**
- **GetTimeStepLength()** : For exporting the length of a single time step **(required)**
- **GetNumberOfTimeSteps()** : For exporting the number of time steps for a complete simulation **(required)**

Steps of Migration (7)

Create a simple reservoir model (SRM) :

{Engine Core after the transformation : Building the .dll}

Functions

- **GetInputTime()** : For exporting the exact time of a specific input **(required)**
- **GetCurrentTime()** : For exporting the current time of the simulation **(required)**
- **GetSimulationStartDate()** : For exporting the starting date of the simulation in a specific format (Year/Month/Day Hours/Minutes/Seconds) **(required)**
- **GetMessage()** : For handling errors **(recommended)**
- **GetNumberOfMessages()** : For handling errors **(recommended)**
- **GetStorage()** : For exporting the value of the Storage variable **(used in testing)**
- **GetFlow()** : For exporting the value of the Outflow variable **(used in testing)**
- **AddInFlow()** : For putting the Inflow values into the Inflow variable. **This function is used by the OpenMI standard in defining the SetValues() method (required)**

Steps of Migration (8)

Create a simple reservoir model (SRM) :

{Engine Core after the transformation : Building the .dll}

Functions

- **RunSimulation()** : For running a full simulation of the model. This function is not used by the OpenMI standard, but it is necessary if we want to run the model as a stand alone application (independent from OpenMI). **(required)**

Steps of Migration (9)

```
{This function performs a single timestep of the model's simulation
 Storage is the amount of water in the reservoir
 WaterDemand is the flow rate from the reservoir to the next branch of the river
 }

function PerformTimeStep() : Boolean ; cdecl ;
var text, err : ShortString ;

begin
    err := 'Error occured in PerformTimeStep method' ;
 try

    Storage := Storage + (Inflow - WaterDemand)*TimeStep ;

    text := FloatToStr(Storage);
    Writeln(OutputFile, text);
    Counter := Counter + 1 ;

    Result := true ;
 except
    NumberOfErrorMessages := NumberOfErrorMessages + 1 ;
    Errors.Add(err) ;
    Result := false ;
 end;
end;

{This function is used to close files that the engine used(e.g. input/output files)

function Finish() : Boolean ; cdecl ;
var err : ShortString ;

begin
   {code here}
  err := 'Error occured in Finish method' ;
 try
  CloseFile(InputFile);     //Closes the InputFile
  CloseFile(OutputFile);    //Closes the OutputFile

  Result := true ;
 except
    NumberOfErrorMessages := NumberOfErrorMessages + 1 ;
    Errors.Add(err) ;
    Result := false ;
 end;
end;

{This function is used for cleaning up/de-allocating memory}

function Dispose() : Boolean ; cdecl ;
```

Steps of Migration (10)

Create the .NET Assemblies :

{Wrapper Classes}

- **SimpleReservoirModelEngineDllAccess class** : It has access to the .dll and makes a one-to-one conversion of all exported functions to public .NET methods
- **SimpleReservoirModelEngineDotNetAccess class** : It changes the calling conventions to C# conventions and error messages to .NET exceptions
- **SimpleReservoirModelEngineWrapper class** : It implements the ILinkableEngine Interface as required for the migration
- **SimpleReservoirModelLinkableComponent class** : It is the OpenMI compliant linkable component ready to be accessed by other models

Steps of Migration (11)

Create the .NET Assemblies :

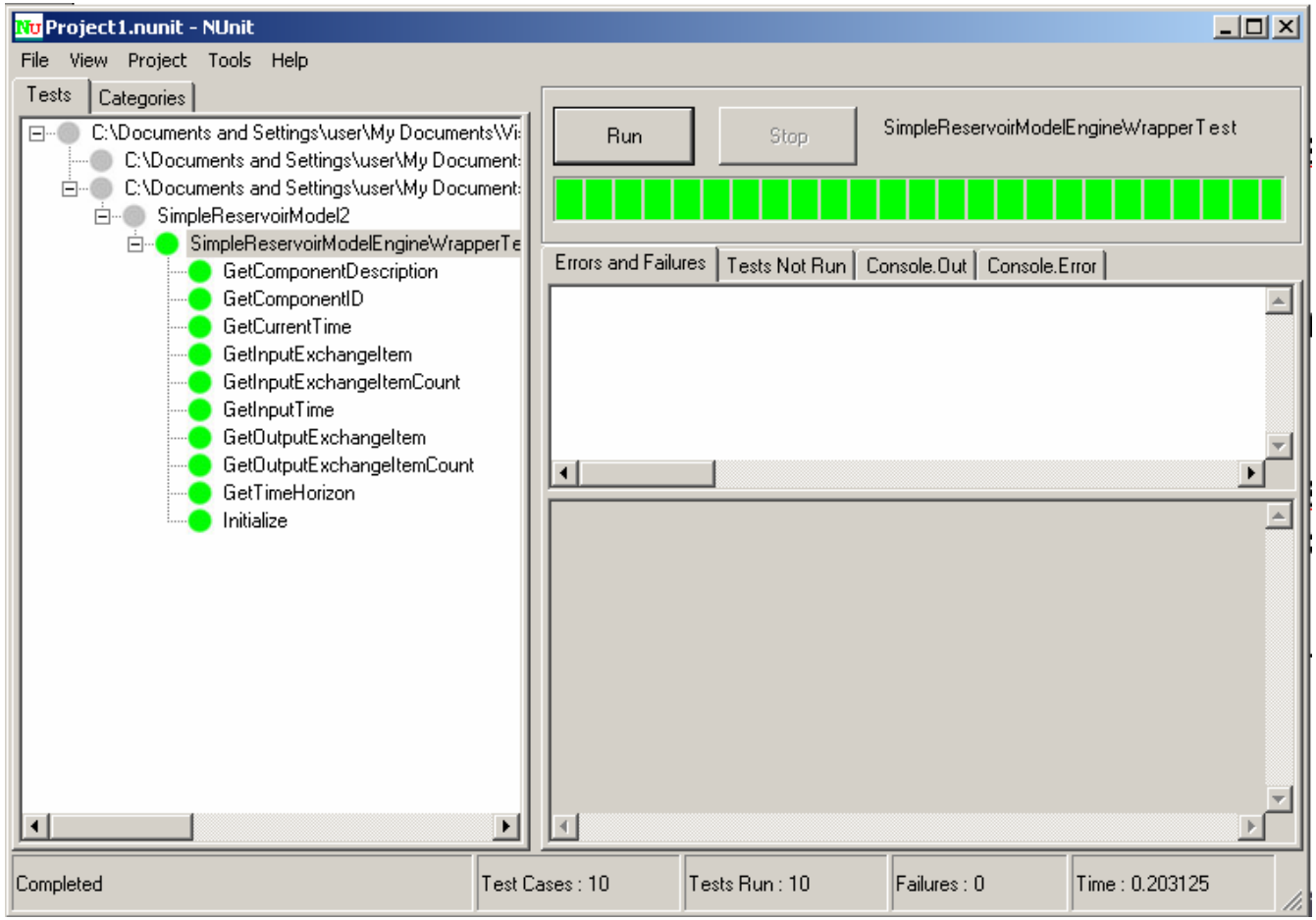
{Test Classes}

- **SimpleReservoirModelEngineDotNetAccessTest class** : For testing the first two classes mentioned before
- **SimpleReservoirModelEngineWrapperTest class** : For testing the SimpleReservoirModelEngineWrapper class
- **SimpleReservoirModelLinkableComponentTest class** : For testing the SimpleReservoirModelLinkableComponent class
- **UseCaseTest class** : For testing the Use Case mentioned in the begining

Steps of Migration (12)

Create the .NET Assemblies :

- Testing Classes Using the NUnit GUI



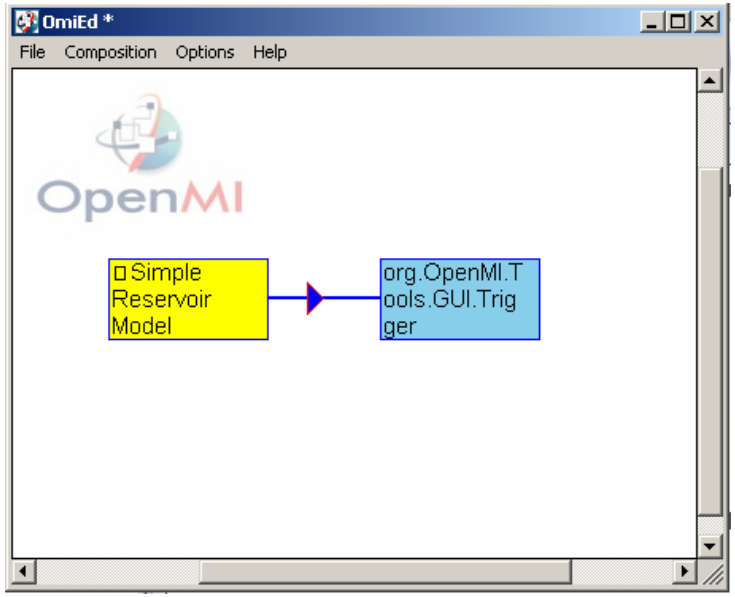
Steps of Migration (13)

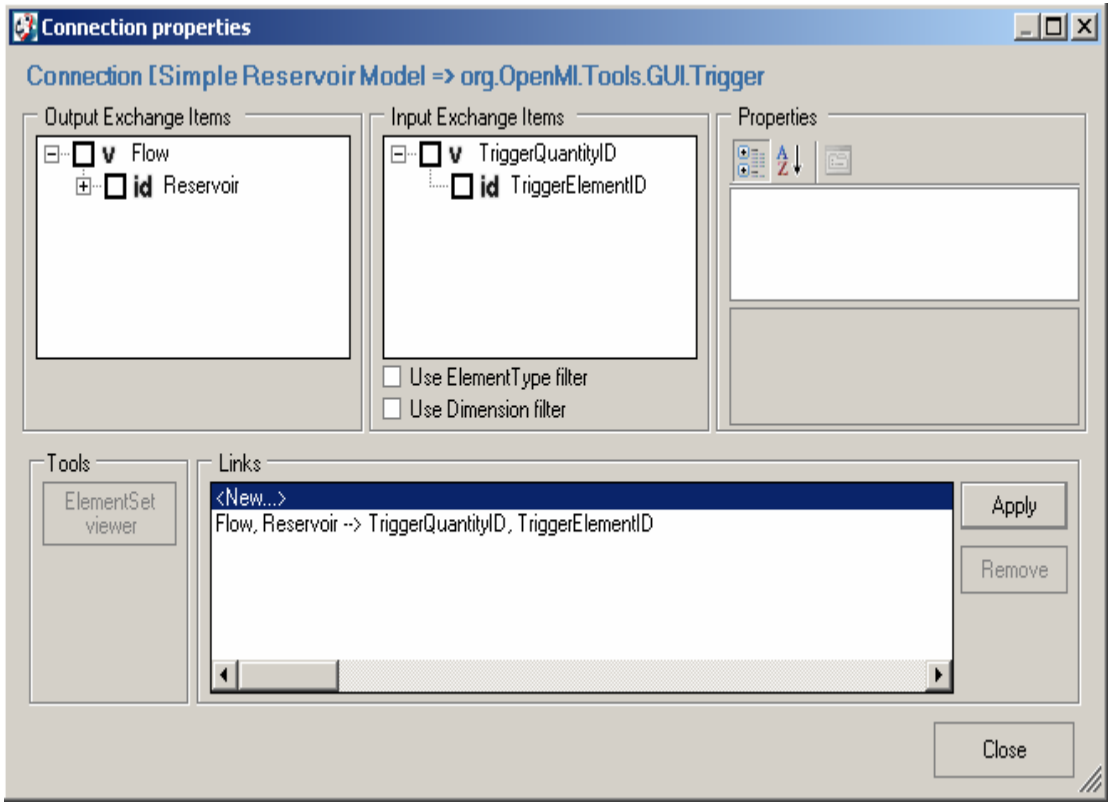
Create the .omi file :

```
<?xml version="1.0" encoding="utf-8" ?>
<LinkableComponent
   Type="SimpleReservoirModel4.SimpleReservoirModelLin
   kableComponent"
   Assembly="SimpleReservoirModel4.dll"> //in the same dir
  <Arguments>
   <Argument Key="FilePath" ReadOnly="true" Value="" />
  </Arguments>
</LinkableComponent>
```

Steps of Migration (14)

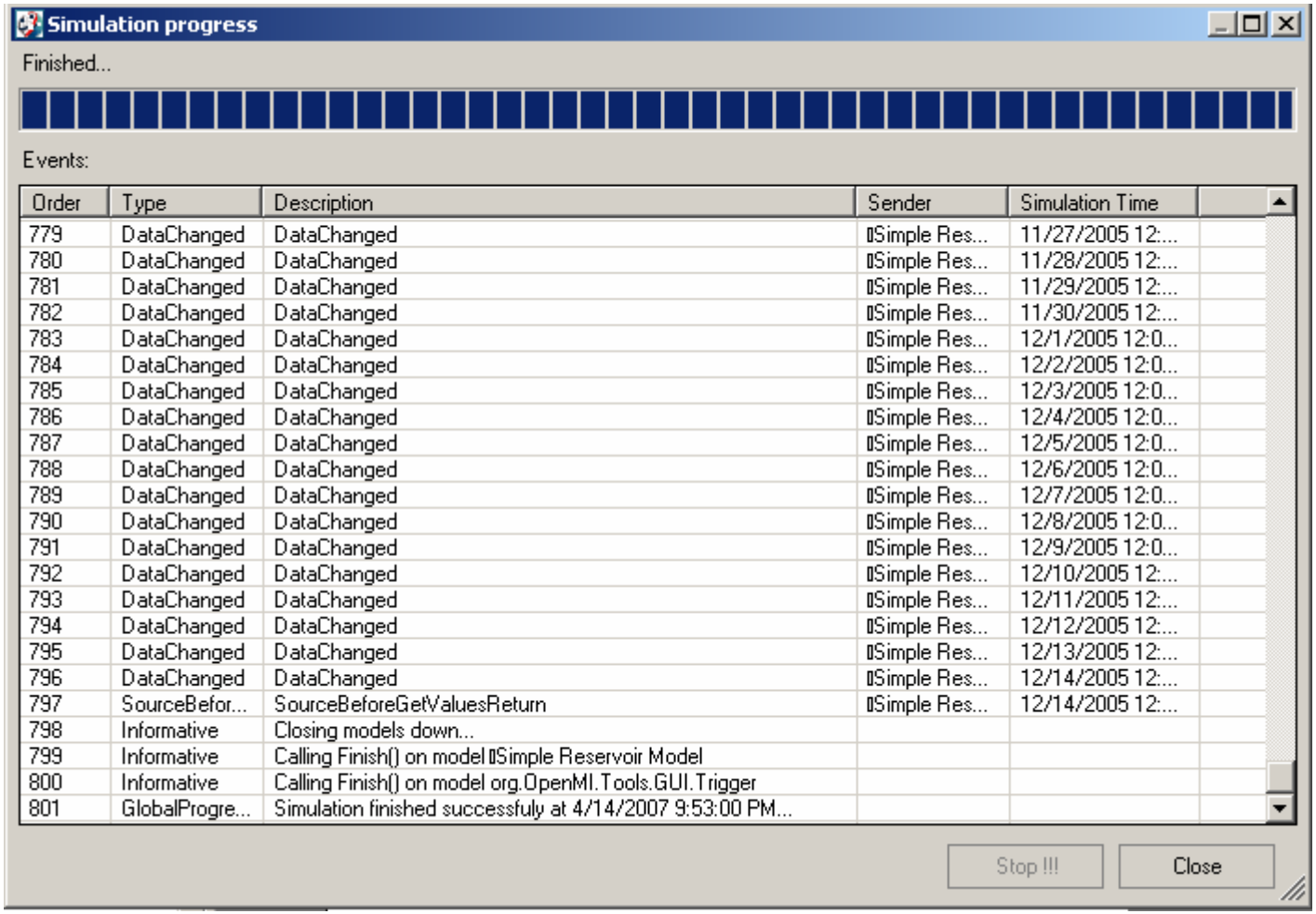
Loading the SRM into the OmiEd :





Steps of Migration (15)

Loading the SRM into the OmiEd :



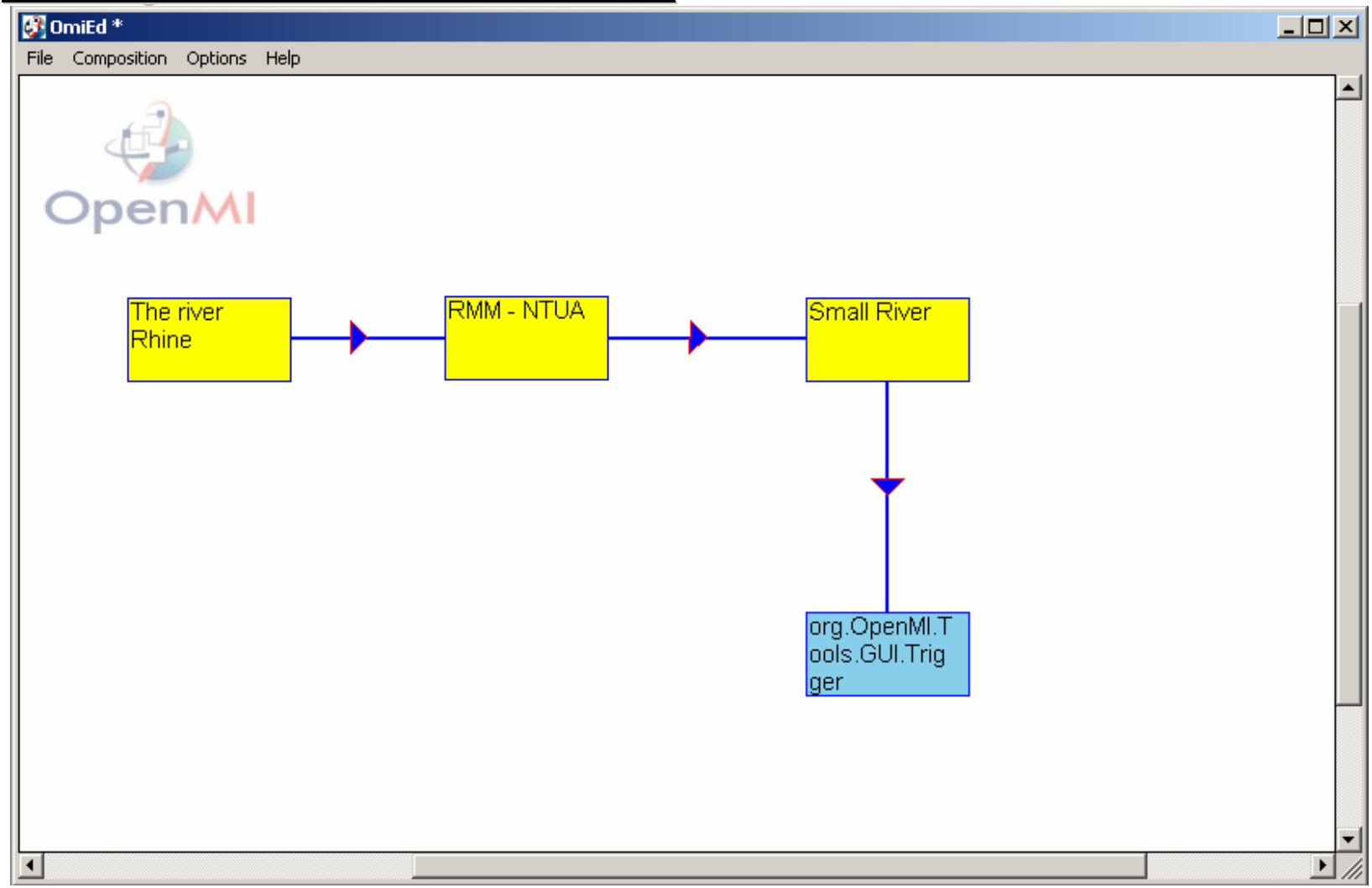
Steps of Migration (16)

Upgrade the initial code :

- **Keep the same structure as defined in the SimpleReservoirEngine.dll**
- **Add new complicated features in each required function mentioned before**
- **Change the ModelID and ModelDescription strings**
- **Change the expected values in Test Classes according to our data**
- **Test the RMM-NTUA with NUnit GUI**
- **Load the RMM onto OmiEd**
- **Run Simulation!!**

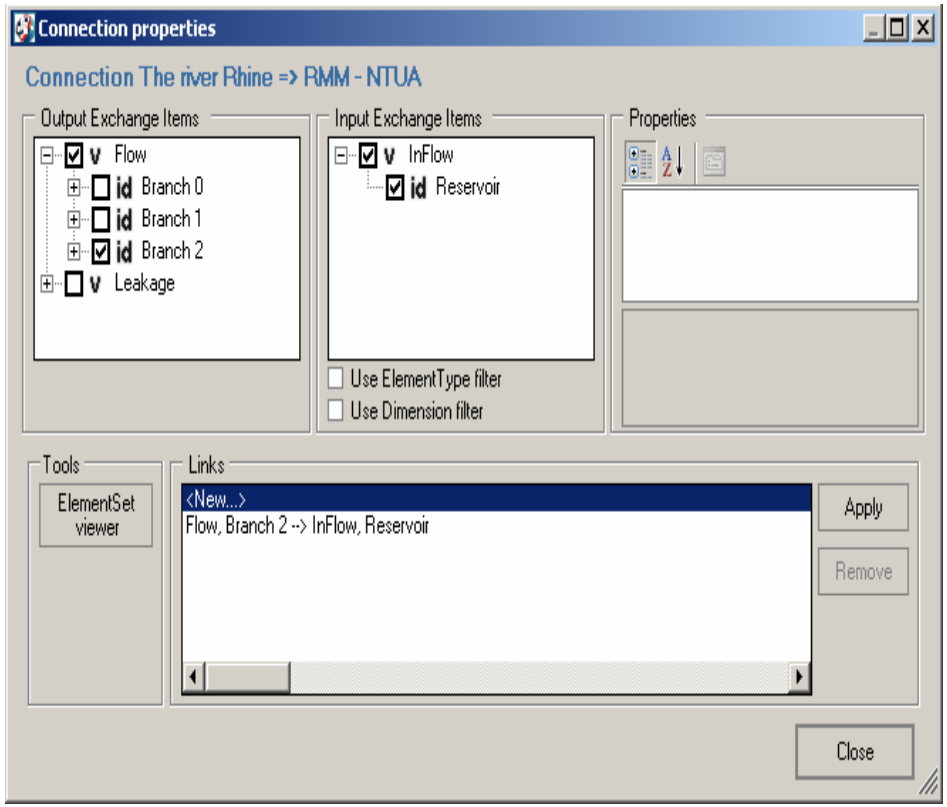
Steps of Migration (17)

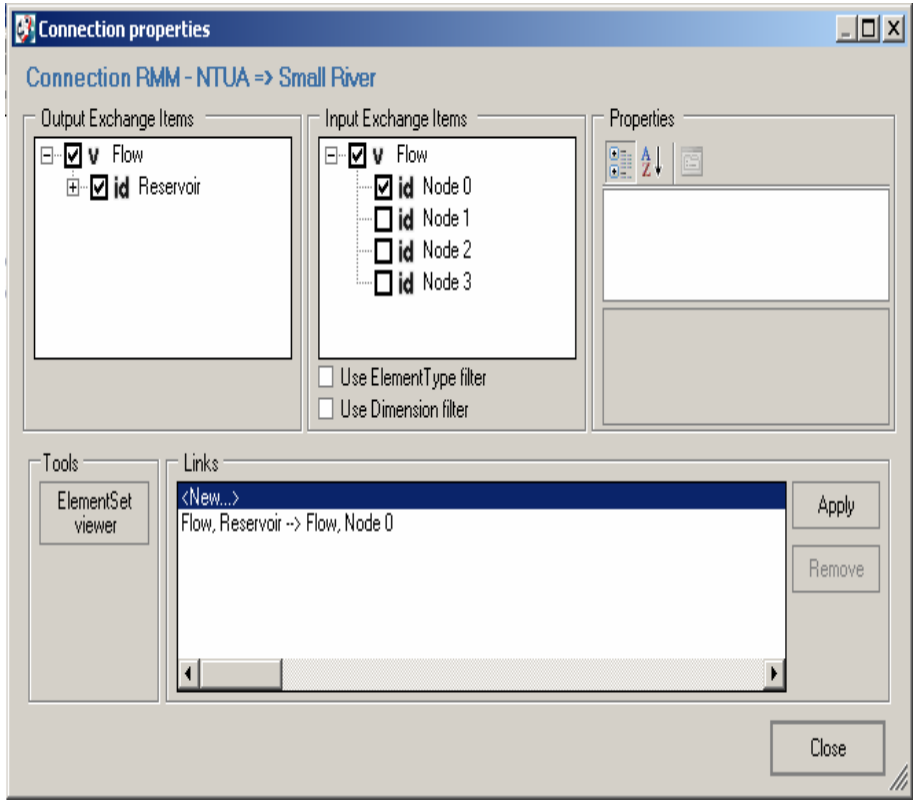
Loading the RMM into the OmiEd :



Steps of Migration (18)

Loading the RMM into the OmiEd :





Steps of Migration (19)

Loading the RMM into the OmiEd :

Simulation progress

Finished...

Events:

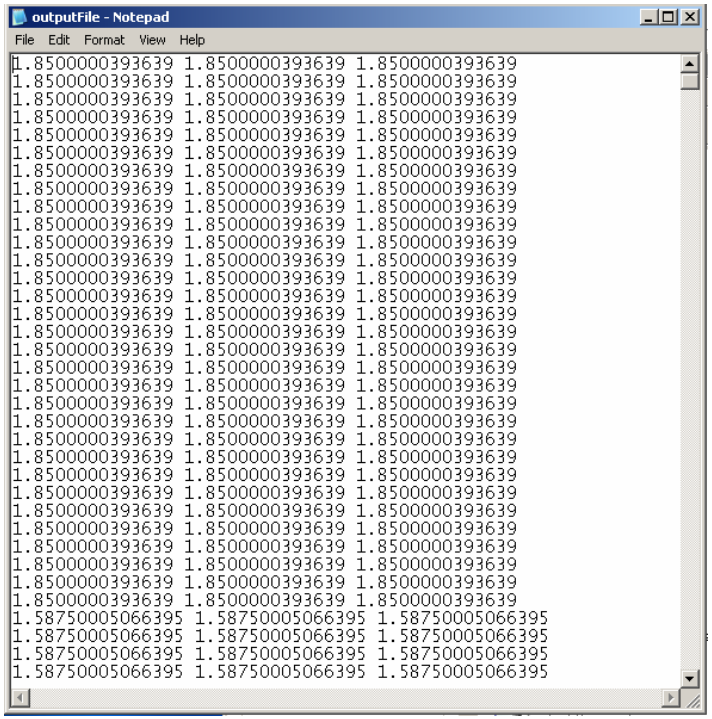
Order	Type	Description	Sender	Simulation Time
3560	DataChanged	DataChanged	RMM - NTUA	12/10/2005 12:...
3561	SourceBefor...	SourceBeforeGetValuesReturn	RMM - NTUA	12/10/2005 12:...
3562	TargetAfterG...	TargetAfterGetValuesReturn	Small River	12/10/2005 12:...
3563	DataChanged	DataChanged	Small River	12/11/2005 12:...
3564	TargetBefor...	TargetBeforeGetValuesCall	Small River	12/11/2005 12:...
3565	SourceAfter...	GetValues(t = 12/11/2005 12:00:00 AM,) <<<===	RMM - NTUA	12/10/2005 12:...
3566	TargetBefor...	TargetBeforeGetValuesCall	RMM - NTUA	12/10/2005 12:...
3567	SourceAfter...	GetValues(t = 12/11/2005 12:00:00 AM,) <<<===	The river Rhi...	12/10/2005 12:...
3568	DataChanged	DataChanged	The river Rhi...	12/11/2005 12:...
3569	SourceBefor...	SourceBeforeGetValuesReturn	The river Rhi...	12/11/2005 12:...
3570	TargetAfterG...	TargetAfterGetValuesReturn	RMM - NTUA	12/10/2005 12:...
3571	DataChanged	DataChanged	RMM - NTUA	12/11/2005 12:...
3572	SourceBefor...	SourceBeforeGetValuesReturn	RMM - NTUA	12/11/2005 12:...
3573	TargetAfterG...	TargetAfterGetValuesReturn	Small River	12/11/2005 12:...
3574	DataChanged	DataChanged	Small River	12/12/2005 12:...
3575	TargetBefor...	TargetBeforeGetValuesCall	Small River	12/12/2005 12:...
3576	SourceAfter...	GetValues(t = 12/12/2005 12:00:00 AM,) <<<===	RMM - NTUA	12/11/2005 12:...
3577	TargetBefor...	TargetBeforeGetValuesCall	RMM - NTUA	12/11/2005 12:...
3578	SourceAfter...	GetValues(t = 12/12/2005 12:00:00 AM,) <<<===	The river Rhi...	12/11/2005 12:...
3579	DataChanged	DataChanged	The river Rhi...	12/12/2005 12:...
3580	SourceBefor...	SourceBeforeGetValuesReturn	The river Rhi...	12/12/2005 12:...
3581	TargetAfterG...	TargetAfterGetValuesReturn	RMM - NTUA	12/11/2005 12:...
3582	DataChanged	DataChanged	RMM - NTUA	12/12/2005 12:...
3583	SourceBefor...	SourceBeforeGetValuesReturn	RMM - NTUA	12/12/2005 12:...
3584	TargetAfterG...	TargetAfterGetValuesReturn	Small River	12/12/2005 12:...
3585	DataChanged	DataChanged	Small River	12/13/2005 12:...
3586	TargetBefor...	TargetBeforeGetValuesCall	Small River	12/13/2005 12:...
3587	SourceAfter...	GetValues(t = 12/13/2005 12:00:00 AM,) <<<===	RMM - NTUA	12/12/2005 12:...
3588	TargetBefor...	TargetBeforeGetValuesCall	RMM - NTUA	12/12/2005 12:...
3589	SourceAfter...	GetValues(t = 12/13/2005 12:00:00 AM,) <<<===	The river Rhi...	12/12/2005 12:...
3590	DataChanged	DataChanged	The river Rhi...	12/13/2005 12:...
3591	SourceBefor...	SourceBeforeGetValuesReturn	The river Rhi...	12/13/2005 12:...
3592	TargetAfterG...	TargetAfterGetValuesReturn	RMM - NTUA	12/12/2005 12:...
3593	DataChanged	DataChanged	RMM - NTUA	12/13/2005 12:...
3594	SourceBefor...	SourceBeforeGetValuesReturn	RMM - NTUA	12/13/2005 12:...
3595	TargetAfterG...	TargetAfterGetValuesReturn	Small River	12/13/2005 12:...
3596	DataChanged	DataChanged	Small River	12/14/2005 12:...
3597	SourceBefor...	SourceBeforeGetValuesReturn	Small River	12/14/2005 12:...
3598	Informative	Closing models down...		
3599	Informative	Calling Finish() on model RMM - NTUA		
3600	Informative	Calling Finish() on model The river Rhine		
3601	Informative	Calling Finish() on model org.OpenMI.Tools.GUI.Trigger		
3602	Informative	Calling Finish() on model Small River		
3603	GlobalProgre...	Simulation finished successfuly at 4/15/2007 11:06:47 AM...		

Stop !!! Close

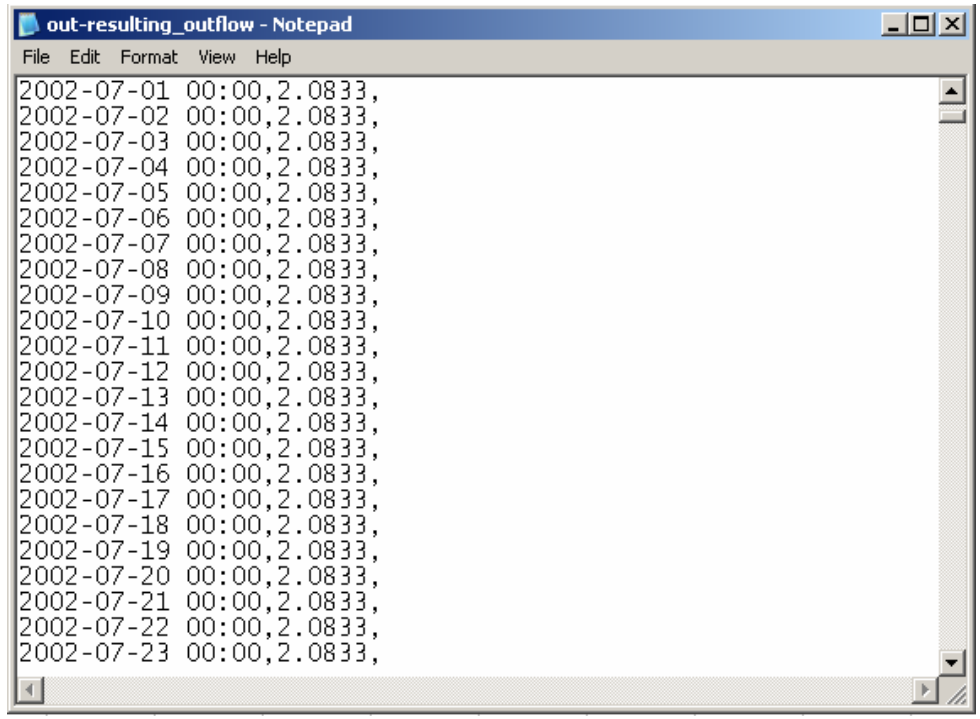
Steps of Migration (20)

Loading the RMM into the OmiEd :

Small River Output File



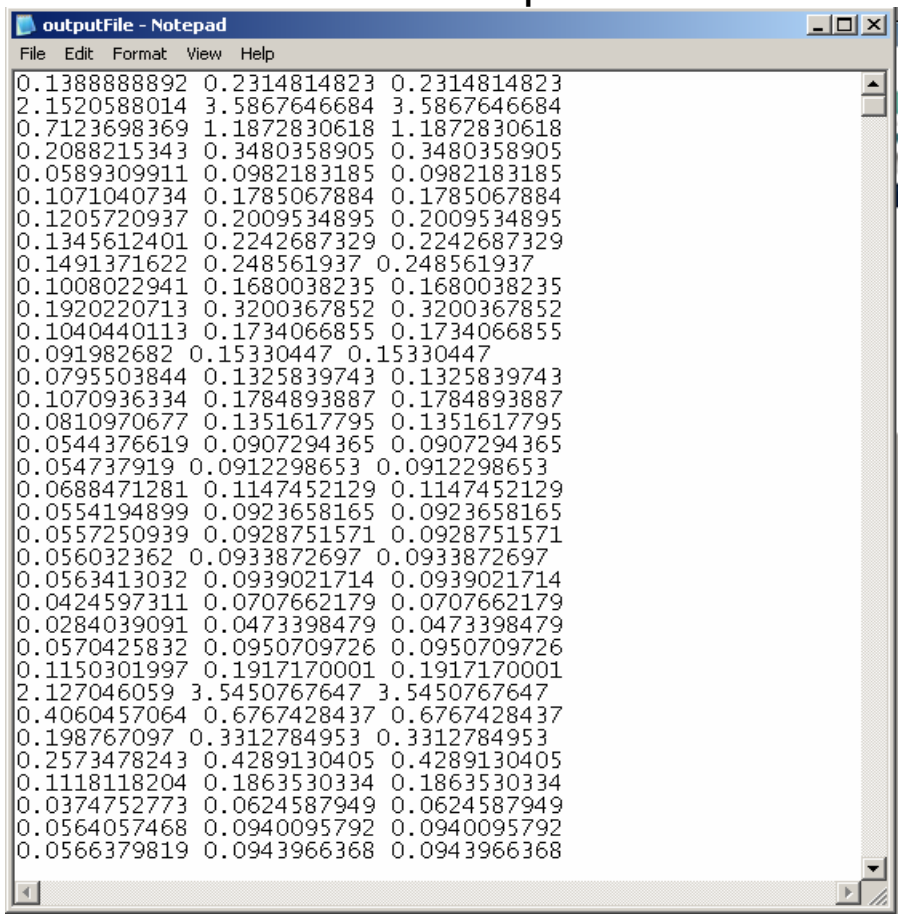
RMM Outflow File



Steps of Migration (21)

Loading the RMM into the OmiEd :

River Rhine Output File

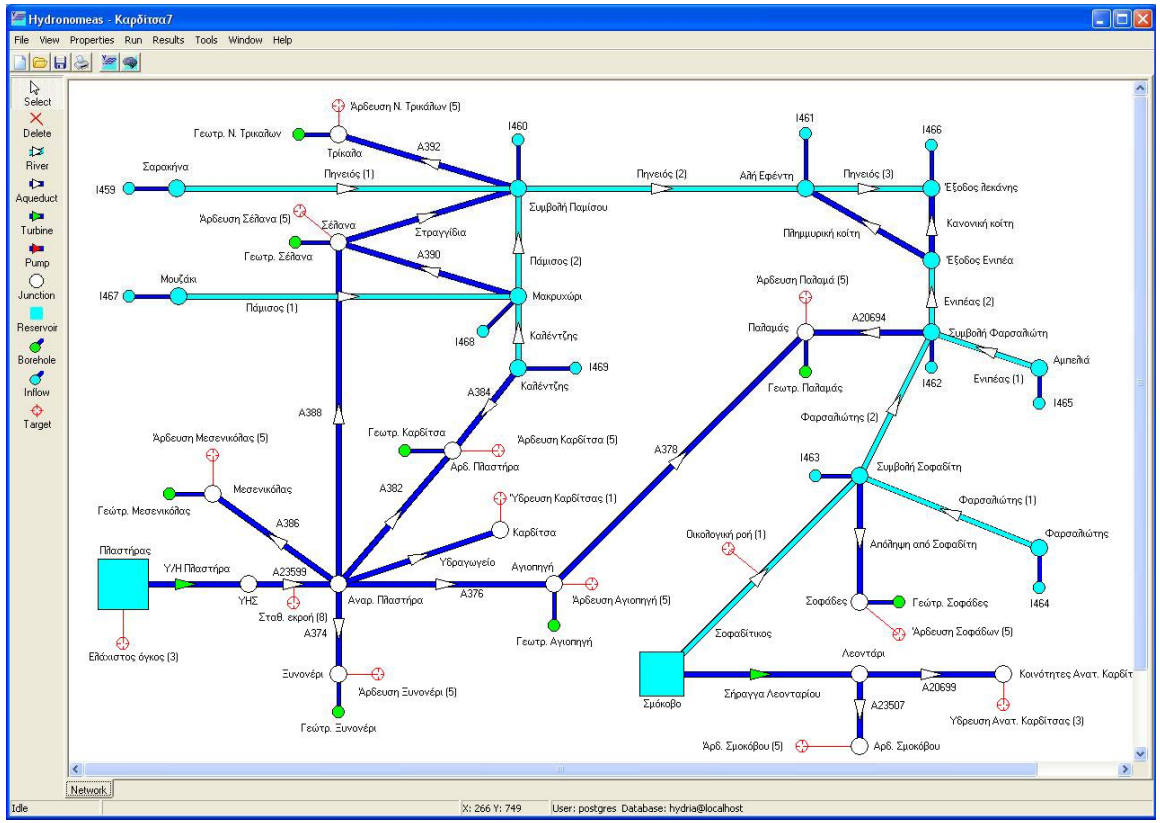


Next Steps

- Link reservoir model RMM-NTUA with MIKE 11.
- Conduct hydrological simulation of the upstream basin using the NAM module (calibration with daily rainfall and discharge data).
- Examine combined hydrological and water management scenarios using the OpenMI platform.
- Account for climate change impacts by modifying precipitation and evapotranspiration data (= model inputs).
- Optimize the reservoir operation rules, on the basis of maximizing reliability.

Future Perspectives

- Migrate the DSS **Hydronomeas**, an advanced tool for optimizing the management of complex water resource systems.
- Apply it to the entire Thessaly hydrosystem.



Model description and demo are available at:
http://www.odysseusproject.gr/en