QnD User Basic Manual

August 2013

Agricultural and Biological Engineering department

QnD User Basic Manual

QUESTIONS AND DECISIONS

ÍNDEX

FIGURES INDEX

TABLES INDEX

THE MODEL

Within wicked environmental challenges, problems that exist in the nexus of environmental science and environmental values, neatly and elegantly optimized solutions are difficult to find and rarely accepted by stakeholders. Different role players must explore the challenge adaptively and through viewpoints to contribute to their understanding of the situation and to learn about the dynamics and values of other relevant stakeholders. The *Questions and Decisions* ™ (QnD™) system was created to provide an effective and efficient tool to integrate ecosystem, management, economics and sociopolitical factors into a user-friendly game/model framework. QnD is written in object-oriented Java and can be deployed in stand-alone or web-based (browser-accessed) modes. The QnD model links spatial components within geographic information system (GIS) files to the abiotic (climatic) and biotic interactions that exist in an environmental system. QnD can be used in a rigorous modeling role to mimic system elements obtained from scientific data or it can be used to create a "cartoon" style depiction of the system to promote greater learning and discussion from decision participants (Kiker and Thummalapalli, 2009).

QnD has two primary elements: a simulation engine and a user-friendly graphical interface that allows users to explore various scenarios and management options. The developer configures the attributes and processes of the simulation's objects through input files written in extensible markup language (XML) that QnD converts into Java objects (Kiker and Thummalapalli, 2009). This design allows for iterative development of model components as its users learn more about the system being managed. CComponents, DData and PProcesses objets used as input and output in QnD will be described in the XML files.

CComponents objects

Spatially-explicit areas and non-spatially-explicit areas are specifically represented through two primary CComponent objects, *CSpatialUnit* and *CHabitat*. A *CSpatialUnit* is the basic spatial entity of the QnD system. CSpatialUnits can be linked to one another and have a specific location. A *CSpatialUnit* can have either zero or any number of *CSpatialUnits* connected to them. One or more *CHabitat* objects exist within a *CSpatialUnit* and are not spatially defined, except via the relationship with the "homeSpatialUnit". A *CHabitat* can hold any number of local instantiations of CComponent objects (*CLocalComponents*). These *CLocalComponents* have both relationships with both "home" *CHabitat* and *CSpatialUnit*. With this basic QnD object architecture, both simple and complex designs are possible with both spatial and non-spatial elements.

DData objects

DData objects store all the relevant information for a specific QnD simulation. All *DData* objects are created from the input XML, GIS data files or time series files and represent a composite variable storing a set of double values. Each *DData* has several attribute variables that allow for various calculations. All available attributes are not always used for each *DData* as some data objects may use other attribute features while others do not. For example, a *DData* object that is linked with a time series file (through its DriverLink attribute) may constantly change current values over time while another may represent a static variable in the simulation and may not use any other attributes besides a single parameter value. In addition to the primary SimulationEngine-related objects, several packages exist for various housekeeping and organization functionality.

PProcesses objects

PProcess objects provide all state changes and action within QnD. PProcess objects use DData objects as inputs, provide a calculation or series of calculations and then write the resulting products into output DData objects. PProcesses are designed with constituent sub-processes within them to create a series of processes for more complex interactions. Table 1 shows some of the different types of processes that can be bound together in series within QnD.

TABLA 1. DIFFERENT TYPES OF PROCESSES IN QND

Process Type	Definition/Purpose
PProcesses for Calculation	
PAddValue	Input1 + Input2 + Input3… + Input $_n$ = Output(s)
PSubtractValue	Input1 – Input2 - … … - Input $_n$ = Output(s)
PMultiplyValue	Input1 x Input2 x Input3… Input $_n$ = Output(s)
PDivideValue	Input1 / Input2 / … … Input $_n$ = Output(s)
PExponentialValue	Output = e $^{(Input)}$
PSetValue	Output = Input
PMeanValue	Output = (Input1 + Input2 + Input3… + Input $_n$) / n
PTemporalMeanValue	Output = Σ(Input1$_t$ + Input1$_{t+1}$ + Input1 $_{t+2}$ … + Input1 $_{t+n}$) / t
PTemporalRunningAverage-Value	Output = (Input1$_t$ + Input1$_{t-1}$ + Input1 $_{t-2}$ … + Input1 $_{Input2}$) / Input2
PCalculateCurrentValue	Within a given sub process list, this object calculates the all the sub processes above it to get an updated and current DData object value.
Specialty PProcesses	
PTransfer *	(Input – TransferAmount) & (Output(s) + TransferA-mount)
PRelationship**	Two dimensional input/cause (x axis) is used to interpo-late an output/effect (y axis) value.
PSimpleLookUpTable***	Uses two input data values to choose another value from user-defined table and assign it to an output
Logical Processes	
PIfEquals	(If input = output) is not true, then it stops executing any further sub-processes and jumps to the next process in the list
PIfGreaterThan	(If input > output) is not true, then it stops executing any further sub-processes and jumps to the next process in the list
PIfLessThan	(If input < output) is not true, then it stops executing any further sub-processes and jumps to the next process in the list

Simulation Engine Design

The primary element for creating a simulation engine is through the deployment of *component*, *process* and *data* objects. For clarification within QnD designs and labeling, a "C" prefixes Components, a "P" prefixes Precesses, and a "D" prefixes Data objects. *CComponent* objects are the basic items of interest within a simulation. *PProcess* objects provide the action and changes from a state to another. *DData* objects provide the necessary description of various attributes. Objects in QnD are arranged in several packages:

- Components Package: CComponents object defines the DData, CComponents and PProcess storage in Hashmaps, CHabitat object moves the CLocalComponents into the Habitat, CLocalComponent object describes the LocalComponent, CScenario object defines the Scenarios and finally CSpatialUnit object defines the link between components and processes.
- Control Package: Objects here are used mostly in the background and thus do not have the "C, P, D" typology of the SimulationEngine objects. The GameDriver object acts as a main simulator object to coordinate both the GameView and SimulationEngine. The PrimaryGameFrame object provides the main GameView frame. Both of these control objects utilize various factory-style objects (QnDModelCreator and qndMngReader) to read XML input and time series files and to create the various constituent objects.
- Data Package: Objects in this package establish the data and drivers
- Exception Package: This package contains different types of exception objects that QnD can show.
- Processes Package: It contains all the PProcesses objects and SubProcesses.

How QnD runs the processes

The way QnD runs the processes is shows in Figure 1. First it runs "early" processes at the Global scale, then it runs the "early" processes at the Habitat level, then it runs all the LocalComponent processes. Then it runs the "late" processes at the Habitat level, SpatialUnit level and finally at the Global level. All that is done within one time step. This is made to allow both setup (early) and summary (aggregation) processes at a variety of scales (both spatial or temporal), so "late" processes are use for example in order to calculate aggregated or mean values of a certain variable.

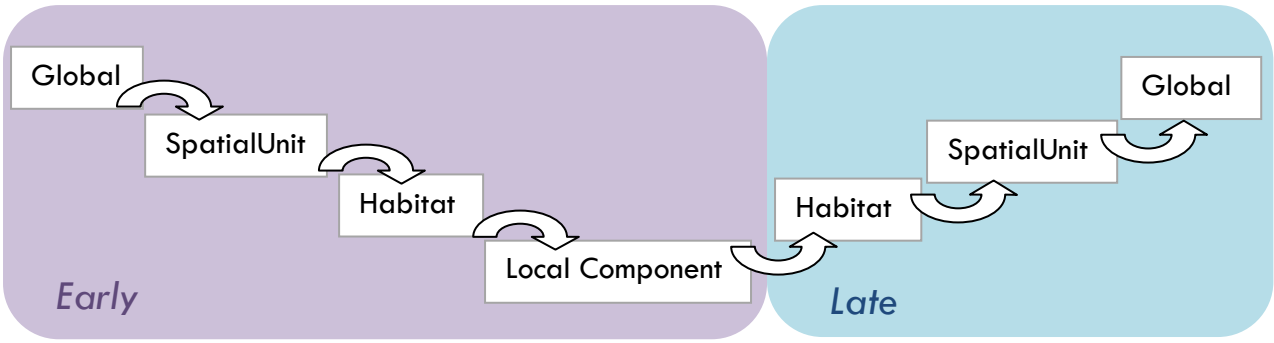


FIGURA 1. SEQUENCY OF HOW QND RUNS THE PROCESSES

EXPLANATION OF THE XML FILES

QnDWorld

It defines the frame of the problem, simply sets up what is going on the world (how many of things and how things are structured). There are no dynamics in this file, only structure. It contains the Global or World DData and the *n* SpatialUnits. Inside each SpatialUnit, some DData and one or several Habitats are defined, and again, inside each Habitat some DData and one or several

CComponents are defined. Finally, each CComponent contains some DData. The structure of the file is the showed in the Figure 2.

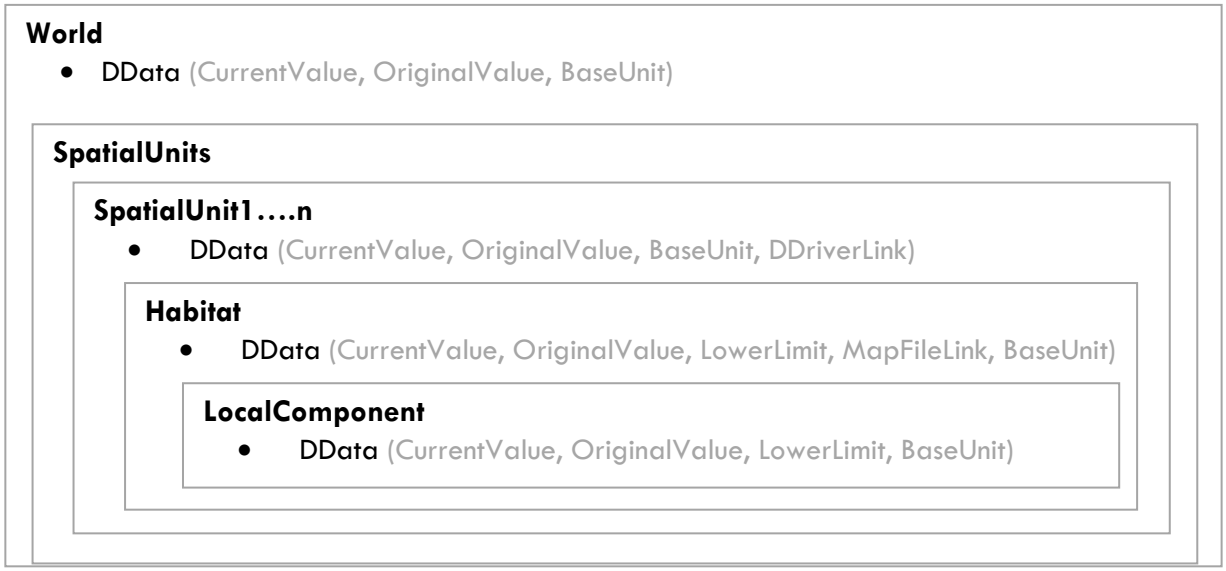


FIGURA 2. STRUCTURE OF LEVELS IN QNDWORLD.XML FILE AND DDATA INFORMATION THAT COULD BE DESCRIBED

QnDComponentDetails

This file sets for each Ccomponents, which are as a hierarchy (CWorld, CSpatialUnit, CHabitat, CLocalComponent), and their DData and PProcesses. The structure of the file is the showed in the Figure 3:

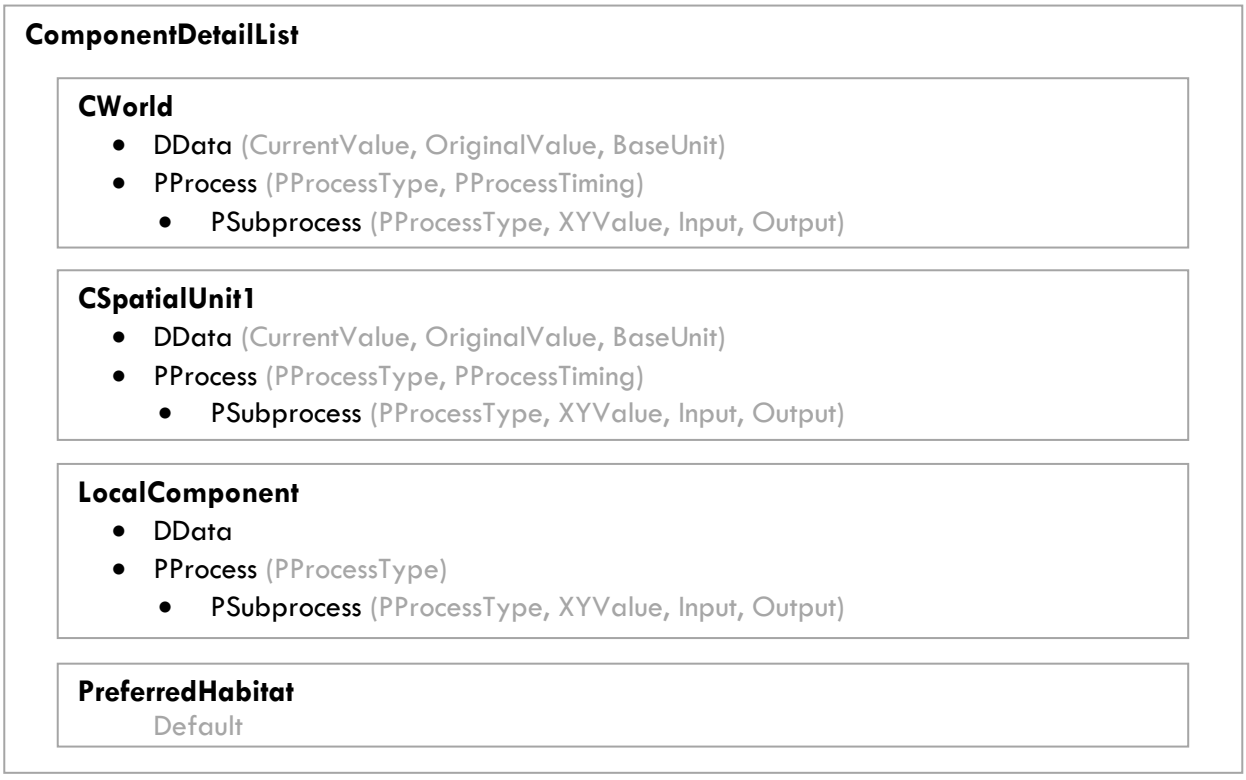


FIGURA 3. STRUCTURE OF LEVELS IN QNDCOMPONENTDETAILS.XML FILE AND DDATA INFORMATION THAT COULD BE DESCRIBED

This XML file is where all the action is. Processes can be defined at global, spatial unit, habitat or localComponent levels. Running time "early" or "later" and process type "PProcess", should be define for each PProcess. Processes comprisse one or more subprocesses. Also subprocess type (relationship, divide, add, multiply, etc.), input and output DData used, need be defined for each subprocess. The subprocess type is defined and implemented in QnD Control package in the simulation engine as PSetValue, PRoundValue, PSubstractValue, PAddValue, etc.

QnDManagement

This file has both time series (date and timed steps used are defined) and external driver (showed in csv input files) configurations along with all the user interface "widget" maps and charts and warning lights (the different ways for GUI are also included here). This can define also the management options and the geometry of the study site.

QnDTopology

It establishes the connection between the spatial units. Any connections can be put here, not just spatial neighbors, even spatial units in another spatial areas can be connected, naming a new "linkage" and then using the description name in various processes. This allows a large amount of freedom beyond simple neighbor operations. However, in simplest cases, connections will be between spatial neighbors.

QnDOutput

This file defines the output files and variables. Any DData in QnD at any level can be output in a comma separated file.

THE QND PROJECT

How run a QnD project from eclipse

1.- First at all, you need to install Eclipse (www.eclipse.org), Europa or Juno, any of them works fine. The use of the latest version, Eclipse Kepler, involves greater difficulty for the use of QnD, so this is not recommended.

2.- Copy all Files to C (at the root path). C:\QnDModel (see Figure 4)

3.- Create a file in C: EclipseProjects.

4.-Open Eclipse.

Select as workspace "C:\EclipseProjects"

4.1- File->New->Project

Java Project

4.2- Project Name: QnD_V1_1

Click on Next

4.3- On Library tab add next Libraries (Click Add External JARs…) (see Figure 5)

4.4- Double click on Control and select GameDriver.java and Run.

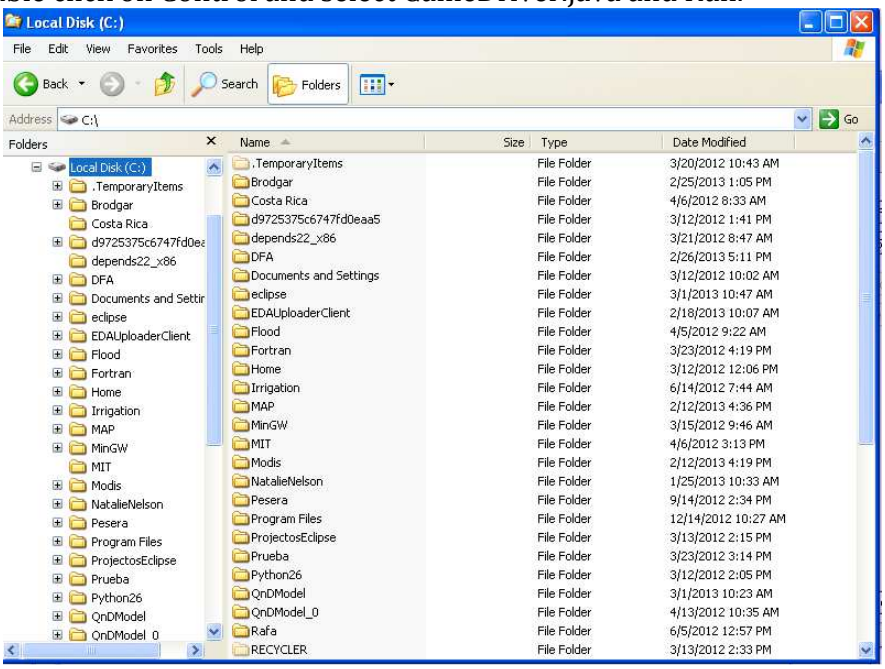


FIGURA 4. SELECTION OF THE ROOT PATH

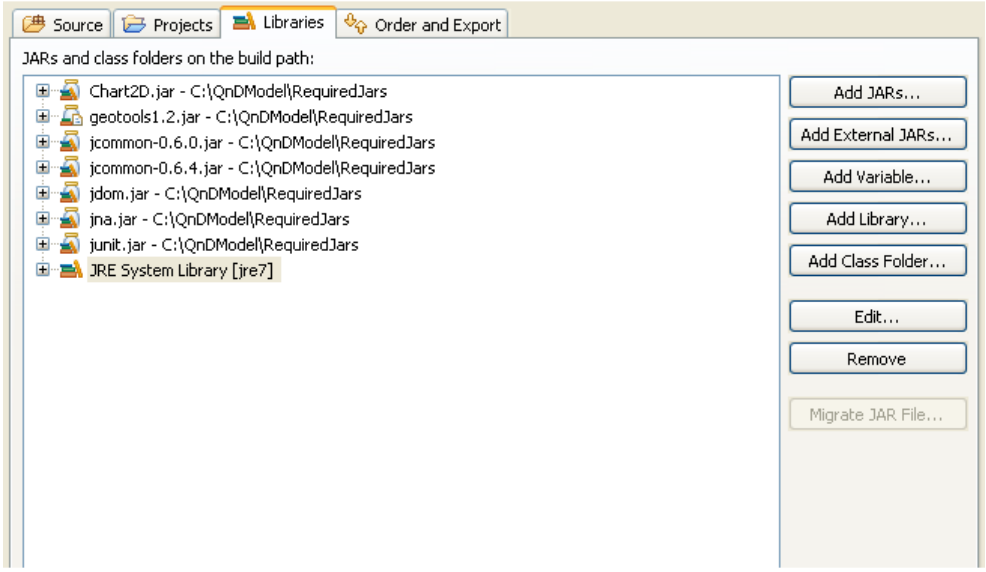


FIGURA 5. ADDITION OF THE LIBRARIES

The QnD GUI

The QnD Graphical User Interface has been designed to select the scenario, time steps and management options of the simulation, as well as display the different outputs. When we run the QnD projects, the GUI window showed in Figure 6 appears.

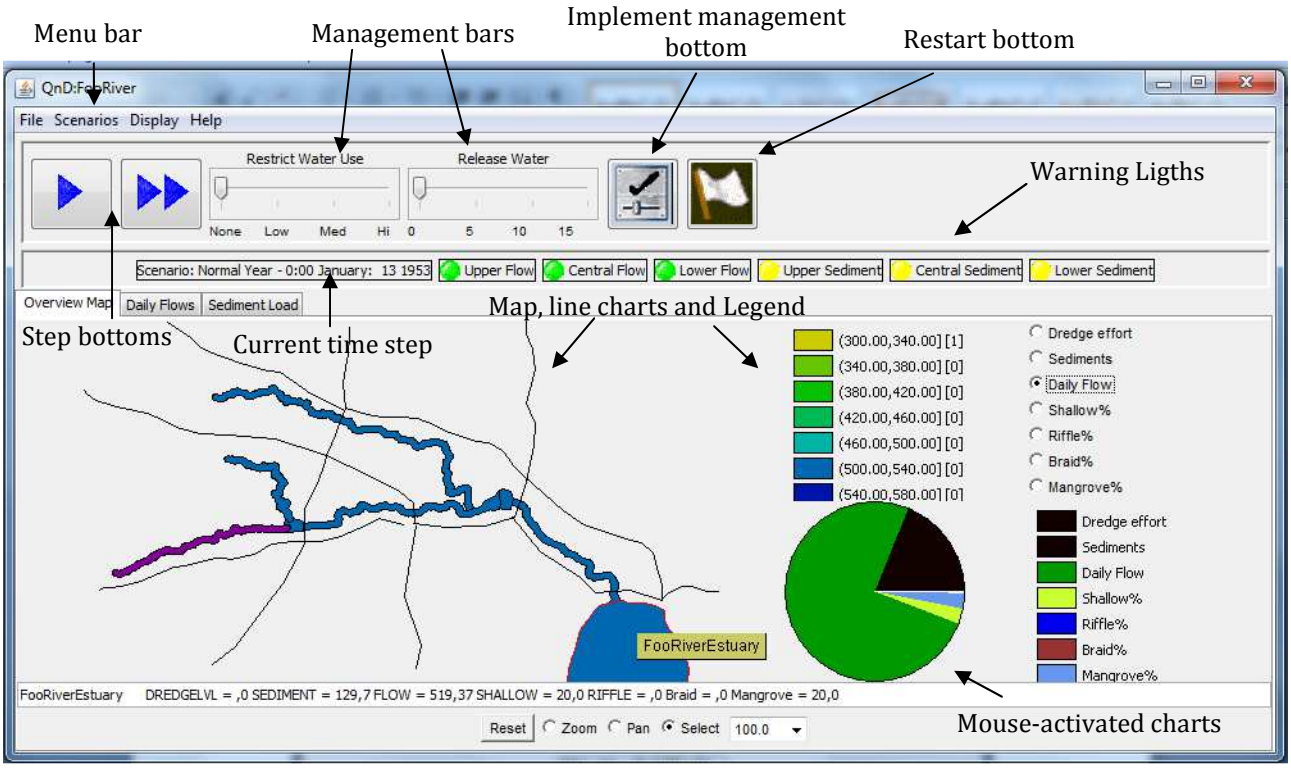


FIGURA 6. VIEW OF THE GUI WINDOW

In this window some bars and bottoms can be found:

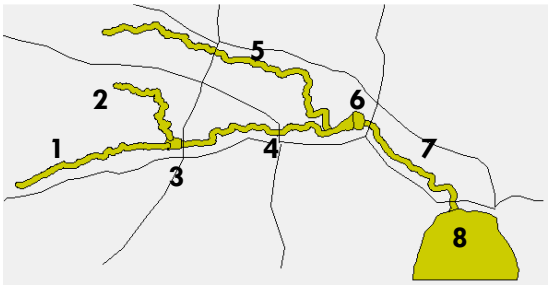
- A menu bar, where previous project can be open or new project can be created, different scenarios can be selected and display options can be changed.
- A management bar, where options management can be changed.
- Implement management and restart bottoms, in order to implement the management settings and restart to the scenario conditions.
- Warning light that change at user-selected critical levels related to a variable. It shows the state of the system at the current time, regarding to a threshold.
- Map, line charts and color legend, which shows DData objects that are spatially explicit. DData objects can be rendered into both collective maps (selected by radio buttons) or line charts.
- Mouse-activated charts and text for individual spatial areas (pie charts and text line descriptions).

SOME SIMPLE EXERCISES: FOOT RIVER EXAMPLE

How is the study site

The first task to do in order to implement QnD in a study site is to prepare the XML files. This task is very important and should be done with extremely attention, because QnD interprets the XML dynamically into a linked object system. Here the FootRiver example is showed.

- Foo river consists of 8 stretches, that are called SpatialUnits (Figure 7):



1. ShakasRapids
2. PetronellaReach
3. FooLockDam2
4. MandelaStraights
5. JoesBend
6. FooLockDam1
7. BobvilleStretch
8. FooRiverEstuary

FIGURA 7. STRETCHES OR SPATIALUNITS IN THE FOOT RIVER

- Also the river receives a daily river flow and annual sediments load (which will be the Drivers of the model).
- In this context, we can do some questions, such as:
    - How is the flow in each stretch?
    - How is the sediment load?
    - How is the fish population?
    - How is the pollutant concentration in the sediment?
    - How it can affect to the benthic organism?
    - How it can affect to fish population?
    - What is the risk level for people?

  In order to answer these questions, some LocalComponents objects, whose processes (Pprocesses and SubProcesses) could be interesting to model, should be defined:

    - CSedimentFooCB: Pollutant concentration in sediment
    - CThinLayerCap: Water layer*
    - CBenthicInvertebrates: Benthic invertebrates
    - CFish: Fish
    - CTaggedFish: Fish controlled by tag campaign*
    - CFisherPeople: People who eats fish

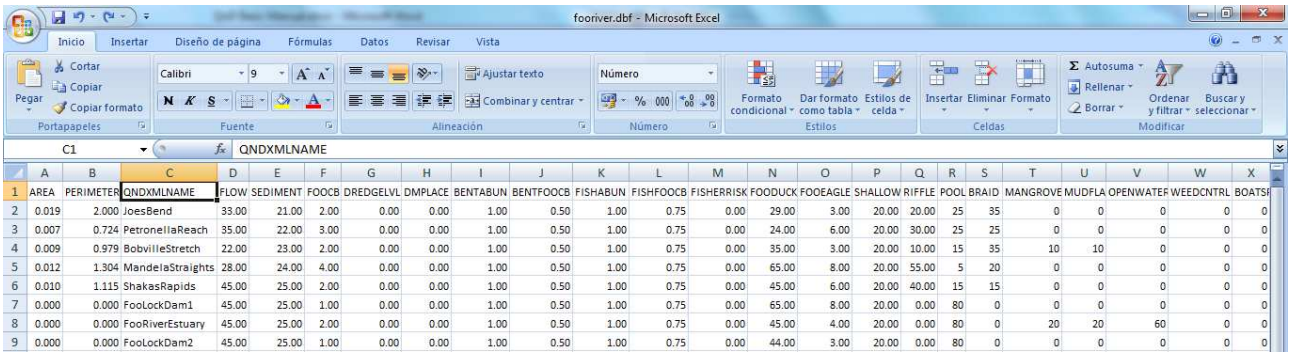
  * CLocalComponent not used in this example, but that could be included in the system, and also processes involved them could be implemented in the model.
- After know how the system (which is called World or Global) works, management decisions, such as control tagged fish or release/restriction decisions regarding to the average load of pollutant could be taken, considering some Global variables, such as financial reserves, management activities and risk index.

The initial conditions

The values of variables or DData to calculate in the first time step have to be defined in the inputs files. This task can be carried out using maps, time series or fix values (original values). In the first case, information is associated to shapefiles. In this example, the shapefile *fooriver* gives the value of some spatial variables for each spatial unit, it can be observed from its attribute table (.dbf file) (Figure 8), setting the value for:

- Area (AREA) and perimeter (PERIMETER)
- The spatial unit (QNDXMLNAME)
- Daily flow (FLOW)
- Sediment concentration (SEDIMENT)
- Pollutant concentration in sediment (FOOCB)
  Level of dredge activities (DREDGELVL)* and dredge material placement (DMPLACE)*
- Abundance of benthic invertebrates (BENTABUN)*
- Pollutant concentration in benthic invertebrates (BENTFOOCB)
- Abundance of fish (FISHABUN)*
- Pollutant concentration in fish (FISHFOOCB)
- Risk for the humans regarding pollutant concentration (FICHERRISK)
- Pollutant concentration in ducks (FOODUCK)*
- Pollutant concentration in eagles (FOOEAGLE)*
- Percentage of shallow (SHALLOW), riffle (RIFFLE), pool (POOL), braid (BRAID) and mangrove (MANGROVE), mud flat (MUDFLAT)* open water (OPENWATER)* area
- Control of weeds (WEEDCNTRL)*
- Boats per month (BOATSPERMO)*

* DData not used in this example, but that could be included in the system and processes involved could be implemented in the model.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	AREA	PERIMETER	QNDXMLNAME	FLOW	SEDIMENT	FOOCB	DREDGELVL	DMPLACE	BENTABUN	BENTFOOCB	FISHABUN	FISHFOOCB	FISHERRISK	FOODUCK	FOOEAGLE	SHALLOW	RIFFLE	POOL	BRAID	MANGROVE	MUDFLA	OPENWATEF	WEEDCNTRL	BOATSI
2	0.019	2.000	JoesBend	33.00	21.00	2.00	0.00	0.00	1.00	0.50	1.00	0.75	0.00	29.00	3.00	20.00	20.00	25	35	0	0	0	0	0
3	0.007	0.724	PetronellaReach	35.00	22.00	3.00	0.00	0.00	1.00	0.50	1.00	0.75	0.00	24.00	6.00	20.00	30.00	25	25	0	0	0	0	0
4	0.009	0.979	BobvilleStretch	22.00	23.00	2.00	0.00	0.00	1.00	0.50	1.00	0.75	0.00	35.00	3.00	20.00	10.00	15	35	10	10	0	0	0
5	0.012	1.304	MandelaStraights	28.00	24.00	4.00	0.00	0.00	1.00	0.50	1.00	0.75	0.00	65.00	8.00	20.00	55.00	5	20	0	0	0	0	0
6	0.010	1.115	ShakasRapids	45.00	25.00	2.00	0.00	0.00	1.00	0.50	1.00	0.75	0.00	45.00	6.00	20.00	40.00	15	15	0	0	0	0	0
7	0.000	0.000	FooLockDam1	45.00	25.00	1.00	0.00	0.00	1.00	0.50	1.00	0.75	0.00	65.00	8.00	20.00	0.00	80	0	0	0	0	0	0
8	0.000	0.000	FooRiverEstuary	45.00	25.00	2.00	0.00	0.00	1.00	0.50	1.00	0.75	0.00	45.00	4.00	20.00	0.00	80	0	20	20	60	0	0
9	0.000	0.000	FooLockDam2	45.00	25.00	1.00	0.00	0.00	1.00	0.50	1.00	0.75	0.00	44.00	3.00	20.00	0.00	80	0	0	0	0	0	0

FIGURA 8. INFORMATION OF THE INITIAL CONDITIONS EN EACH SPATIAL UNIT DEFINED IN THE MAP FILE

Moreover, some initial conditions are set using fix values defined as "current value" in the World.xml file, as Table 2 shows. Note that some variables such as *DPoolPercentArea, DRifflelPercentArea*, etc. are also defined in the map (SHALLOW, RIFFLE, etc.), so they will be update by the shapefile *fooriver.* In the same way, initial conditions of DData at World level are also defined in the World.xml file (see Table 4 later).

TABLA 2. INITIAL CONDITIONS (CURRENT VALUES) OF DDATA DESCRIBED IN EACH SPATIAL UNIT IN THE WORLD.XML FILE

Variable	Spatial units							
	ShakaRapids	**PetronellaReach**	**FooLockDam2**	**MandelaStraights**	**JoesBend**	**FooLockDam1**	**BobvilleStretch**	**FooRiverEstuary**
(CSpatialUnit)								
DArea	10	10	10	10	10	10	10	10
DXCoordinate	3	3	3	3	3	3	3	3
DYCoordinate	0	0	0	0	0	0	0	0
DDailyRiverFlow	0	0	0	0	0	0	0	0
DAnnual SedimentLoad	0	0	0	0	0	0	0	0
DDredgeEffort	0	0	0	0	0	0	0	0
DDredgeMaterialPlacement	0	0	0	0	0	0	0	0
DDredgeEffortCost	0	0	0	0	0	0	0	0
DDredgeMaterialPlacement	0	0	0	0	0	0	0	0
(CHabitat)								
DPercentArea	100	100	100	100	100	100	100	100
DShallowPercentArea	0	0	0	0	0	0	0	0
DPoolPercentArea	0	0	0	0	0	0	0	0
DRiffllePercentArea	0	0	0	0	0	0	0	0
DBraidPercentArea	0	0	0	0	0	0	0	0
DMangrovePercentArea	0	0	0	0	0	0	0	0
(CSedimentFooCB)								
DConcentration	1	8	3	1	2	4	1	1
(CThinLayerCap)								
DDepth	0	0	0	0	0	0	0	0
(CBenthicInvertebrates)								
DAbundance	1	1	1	1	1	1	1	1
DFooCBConcentration	1	1	1	1	1	1	1	1
(CFish)								
DAbundance	1	1	1	1	1	1	1	1
DFooCBConcentration	1	1	1	1	1	1	1	1
DRisk	0	0	0	0	0	0	0	0
DPopulation	100	50	60	70	85	90	45	65
(CFisherPeople)								
DRisk	0	0	0	0	0	0	0	0
(CTaggedFish)								
DPopulation	1	1	1	1	1	1	1	1
DConditionIndex	1	1	1	1	1	1	1	1

Preparation of the Topology.xml file

First of all, the structure of the system should be defined, this means how the SpatialUnits are connected. The links between SpatialUnits are described in this file by the spatial situation of the link, as "SpatialNeighbor", "DownStream" and "Upstream", as it is showed in Table 3.

TABLA 3. DESCRIPTION OF TOPOLOGY

Tag name/Text	=Name	SpatialUnit	SpatialUnit	SpatialUnitt
HomeSpatialUnit Name	= "FooLockDam2"			
Linkage Name	= "SpatialNeighbor"	ShakasRapids	PetronellaReach	MandelaStraights
Linkage Name	= "DownStream"	MandelaStraights		
Linkage Name	= "UpStream"	ShakasRapids	PetronellaReach	

An example of this structure in XML schema would be:

```
<TopologyList>
        <HomeSpatialUnit Name="FooLockDam2">
                <Linkage Name = "SpatialNeighbor">
                        <SpatialUnit>ShakasRapids</SpatialUnit>
                        <SpatialUnit>PetronellaReach</SpatialUnit>
                        <SpatialUnit>MandelaStraights</SpatialUnit>
                </Linkage>
                <Linkage Name = "DownStream">
                        <SpatialUnit>MandelaStraights</SpatialUnit>
                </Linkage>
                <Linkage Name = "UpStream">
                        <SpatialUnit>ShakasRapids</SpatialUnit>
                        <SpatialUnit>PetronellaReach</SpatialUnit>
                </Linkage>
        </HomeSpatialUnit>
        *And the same for another HomeSpatialUnits….
```

Preparation of the World.xml file

All the Spatial Units, Habitat and LocalComponent, together with their respective DData, should be described in this file. An example of tag and subtag that can be defined in the World.xml file in the FootRiver example are showed here.

- In World level, 3 DData are defined (*DFinancialReserves, DManagementPopularityIndex, DOverallRiskIndex*), as it is showed in Table 4. They are general DData, which affect to the global system and that will be used for its management. Some characteristics regarding DData, such as base units, original, current and limit values, can be defined here.

TABLA 4. DDATA AND SPATIALUNITS DESCRIBED IN WORLD LEVEL

Tag name/Text	=Name	CurrentValue	OriginalValue	BaseUnit	CSpatialUnit
DData	DFinancialReserves	10.00	10.00	USDollars	
DData	DManagementPopularityIndex	1.00	1.00	none	
DData	DOverallRiskIndex	1.00	1.00	none	
CSpatialUnits					CSpatialUnit (8)

- Also, inside World level 8 Spatial Units are defined (Table 5), and for each one DData and CHabitat objects. Note that in this example there is only one Habitat in each SpatialUnit.

TABLA 5. DDATA AND SPATIALUNITS DESCRIBED IN SPATIALUNIT LEVEL

Tag name/Text	=Name	DData	CHabitat
CSpatialUnit	ShakasRapids	DData (9)	CHabitat
CSpatialUnit	PetronellaReach	DData (9)	CHabitat
CSpatialUnit	FooLockDam2	DData (9)	CHabitat
CSpatialUnit	MandelaStraights	DData (9)	CHabitat
CSpatialUnit	JoesBend	DData (9)	CHabitat
CSpatialUnit	FooLockDam1	DData (9)	CHabitat
CSpatialUnit	BobvilleStretch	DData (9)	CHabitat
CSpatialUnit	FooRiverEstuary	DData (9)	CHabitat

- Table 6 shows DData and CHabitat objects of the SpatialUnit Shakasrapids. In this case, DData objects are variables that describes that SpatialUnit, such as area, coordinates, daily river flow, annual sediment loads, etc. Also, DriverLinks needs to be defined here. In this example, the DDriverLink is *Simulated River Flow* that will be linked to *DDailyRiverFlow* and *AnnualSedimentLoads* objects, with *River Flow* and *Sediment Loads* DDataLinks, respectively.

- Moreover, a Habitat (or more than one) could be defined in the SpatialUnit, that contains DData objects (the area percent of different kind waters), and LocalComponent objects (C*sedimentFooCB*, *CBenthicinvertebrates, CFish*, etc.), as Table 7 shows. In Habitat level, the percentage of shallow, pool, riffle, braid and manglobe area is described, so a MapFileLink have to be defined for each one (Table 7). MapFileLinks are references to GIS files where information about DData is showed in the dbf file. In this file there is a field named QNDXMLNAME that shows the SpatialUnits (Figure 8). Finally, DData for each LocalComponent are also defined (Table 8).

TABLA 6. DDATA AND CHABITAT DESCRIBED IN SPATIALUNIT SHAKASRAPIDS LEVEL

Tag name/Text	=Name	Current Value	Original Value	BaseUnit	DDriverLink	DDataLink	DData	CLocal Component
DData	DArea	10.00	10.00	sq km				
DData	DXCoordinate	3.00	3.00	RiverMile				
DData	DYCoordinate	0.00	0.00	Meters				
DData	DDailyRiverFlow	0.00	0.00	Meters	Sim. River Flow	River Flow		
DData	DAnnualSedimentLoad	0.00	0.00	None	Sim. River Flow	Sediment Load		
DData	DDredgeEffort	0.00	0.00	None				
DData	DDredgeMaterialPlacement	0.00	0.00	None				
DData	DDredgeEffortCost	0.00	0.00	None				
DData	DDredgeMaterialPlacementCost	0.00	0.00	None				
CHabitat	CHabitat						DData (6)	CLocalComponent (6)

TABLA 7. DDATA AND CLOCALCOMPONENT DESCRIBED IN HABITAT LEVEL

Tag name/Text	=Name	Current Value	Original Value	LowerLimit	MapFileLink	BaseUnit	DData
DData	DPercentArea	100.00					
DData	DShallowPercentArea	0.00	0.00		SHALLOW	percentArea	
DData	DPoolPercentArea	0.00	0.00	0.00	POOL	percentArea	
DData	DRiffllePercentArea	0.00	0.00	0.00	RIFFLE	percentArea	
DData	DBraidPercentArea	0.00	0.00	0.00	BRAID	percentArea	
DData	DMangrovePercentArea	0.00	0.00	0.00	MANGROVE	percentArea	
CLocalComponent	CSedimentFooCB						DData
CLocalComponent	DConcentration						DData
CLocalComponent	CThinLayerCap						DData
CLocalComponent	DDepth						DData
CLocalComponent	CBenthicInvertebrates						DData (2)
CLocalComponent	CFish						DData (4)
CLocalComponent	CTaggedFish						DData (2)
CLocalComponent	CFisherPeople						DData

TABLA 8. CLOCALCOMPONENT AND DDATA DESCRIBED IN EACH CLOCALCOMPONENT LEVEL

Name	Value	Tag name/Text	=Name	Current Value	Original Value	UpperLimit	LowerLimit	BaseUnit
Name	CSedimentFooCB	DData	DConcentration	1.0	1.0		0.00	partsPerMillion
Name	CThinLayerCap	DData	DDepth	0.0	0.0		0.00	cm
Name	CBenthicinvertebrates	DData	DAbundance	1.0	1.0		0.00	none
		DData	DFoodConcentration	1.0	1.0		0.00	partsPerMillion
Name	CFish	DData	DAbundance	1.0	1.0	1.00	0.00	none
		DData	DFoodConcentration	1.0	1.0		0.00	partsPerMillion
		DData	DRisk	0.0	0.0		0.00	none
		DData	DPopulation	100.0	100.0			fish
Name	CTaggedFish	DData	DPopulation	1.0	1.0			Fish
Name	CFisherPeople	DData	DRisk	0.0	0.0		0.00	none

An example of this structure in XML schema would be:

```
CWorld>
 <DData Name = "DFinancialReserves">
    <CurrentValue Value="10.00"> </CurrentValue>
    <OriginalValue Value="10.00"> </OriginalValue>
    <BaseUnit Name = "USDollars"> </BaseUnit>
 </DData>
 *And the same for other DData in the World level….

 <CSpatialUnits>

     <CSpatialUnit Name="ShakasRapids">
         <DData Name = "DArea">
                 <CurrentValue Value="10.00"> </CurrentValue>
                 <OriginalValue Value="10.00"> </OriginalValue>
                 <BaseUnit Name = "sq km"> </BaseUnit>
         </DData>
         <DData Name = "DDailyRiverFlow">
                 <CurrentValue Value="0.00"> </CurrentValue>
                 <OriginalValue Value="0.00"> </OriginalValue>
                 <DDriverLink Name="Simulated River Flow">
                         <DDataLink Name="River Flow2"> </DDataLink>
                 </DDriverLink>
         </DData>
         *And the same for other DData in the SpatialUnit….

         <CHabitat Name = "Default">
                 <DData Name="DPercentArea">>
                         <CurrentValue Value="100.00"> </CurrentValue>
                 </DData>
                 <DData Name="DShallowPercentArea">
                         <CurrentValue Value="0.0"> </CurrentValue>
                         <OriginalValue Value="0.0"> </OriginalValue>
                         <LowerLimit Value = "0.00"> </LowerLimit>
                         <MapFileLink Name ="SHALLOW"> </MapFileLink>
                         <BaseUnit Name = "percentArea"> </BaseUnit>
                 </DData>
                 *And the same for other DData in the Habitat….

                 <CLocalComponent Name = "CSedimentFooCB">
                         <DData Name="DConcentration">
                                 <CurrentValue Value="1.0"> </CurrentValue>
                                 <OriginalValue Value="1.0"> </OriginalValue>
                                 <LowerLimit Value = "0.00"> </LowerLimit>
                                 <BaseUnit Name = "partsPerMillion"> </BaseUnit>
                         </DData>
                         *And the same for other DData in the LocalComponent….
                 </CLocalComponent>

                 <CLocalComponent Name = "CBenthicInvertebrates">
                         <DData Name="DAbundance">
                                 <CurrentValue Value="1.0"> </CurrentValue>
                                 <OriginalValue Value="1.0"> </OriginalValue>
                                 <UpperLimit Value = "1.00"> </UpperLimit>
                                 <LowerLimit Value = "0.00"> </LowerLimit>
                                 <BaseUnit Name = "none"> </BaseUnit>
                                 <MapFileLink Name ="BENTABUN"> </MapFileLink>
```

```
                    </DData>
                    <DData Name="DFooCBConcentration">
                        <CurrentValue Value="1.0"> </CurrentValue>
                        <OriginalValue Value="1.0"> </OriginalValue>
                        <LowerLimit Value = "0.00"> </LowerLimit>
                        <BaseUnit Name = "partsPerMillion"> </BaseUnit>
                    </DData>
                    *And the same for other DData in the LocalComponent….
                </CLocalComponent>
                *And the same for other LocalComponent in the Habitat….
    </CHabitat>
        *And the same for other Habitat in the SpatialUnit….
    </CSpatialUnit>
        *And the same for other SpatialUnit in the World level….
</CWorld>
```

Preparation of the DetailList.xml file

This file is where the action is, where the processes are modeled. In each level, DData and PProcesses, together with their respective SubProcesses have to be described.

All PProcess have to include at least one PSubProcess, and this can be implemented in two different ways:

1. From relationship provided by experts, in this case PProcess Type will be “Relationship” and two o more pairs of XY data will be required.
2. From parametric equations, in this case PProcess Type will be considered as “PMultiply”, “PDivided”, etc.

In every case an input and output DData has to be defined, and also describing their SpatialUnit Name or LocalComponent Name, according the level they belong. Moreover, DData Type “CurrentValue” or “CumulativeEffect” need to be defined, according to the character of the variable calculated.

In this example, 6 LocalComponent are described in all the SpatialUnit (see Table 7).

For example, we are going to show how implement in QnD a parametric equation type process: the growth of fish population. This will be got in only 2 steps:

1. We need to know in what level we are going to implement the process. In this case, as this process refers to a LocalComponent (CFish), we need to implement the process at LocalComponent level. So, in the DetailList.xml file we will add the process like "PFishPopIncrease", describing what type of process is and the subprocesses that it uses (“PFishHaveBabies”), and also subprocess type and inputs (“DPopulation” and "DFishBirthRate")and outputs ("DPopulation") required.
2. Secondly, every DData used particularly in these subprocesses should be defined in the DetailList.xml file, at the suitable LocalComponent level. In this case, DData inputs and outputs are from CFish LocalComponent, so they have to be defined in this level.
3. Finally, DData used in this but also in other processes have to be been defined in the World.xml file at LocalComponent level, defining the base unit, current and original values.

So, the structure in the DetailList.xml file would be in this way:

```
<ComponentDetailList>
<CLocalComponent Name="CFish">

        <DData Name="DFishBioaccumulationFactor">
                <CurrentValue Value="2.3"> </CurrentValue>
                <OriginalValue Value="2.3"> </OriginalValue>
                <BaseUnit Name = "none"> </BaseUnit>
        </DData>
        <DData Name="DFishBirthRate">
                <CurrentValue Value="1.1"> </CurrentValue>
                <OriginalValue Value="1.1"> </OriginalValue>
                <BaseUnit Name = "none"> </BaseUnit>
        </DData>
        <PProcess Name = "PFishPopIncrease"
                PProcessType = "PProcess" >
                <PSubProcess Name = "PFishHaveBabies"
                        PProcessType = "PMultiplyValue" >
                        <Input LocalComponentName="CFish"
                                DData="DPopulation"
                                DataType="CurrentValue">
                        </Input>
                        <Input LocalComponentName="CFish"
                                DData="DFishBirthRate"
                                DataType="CurrentValue">
                        </Input>
                        <Output LocalComponentName="CFish"
                                DData="DPopulation"
                                DataType="CurrentValue">
                        </Output>
                </PSubProcess>
        </PProcess>
</CLocalComponent>
```

And also DPopulation DData will be defined in the World.xml file at the SpatialUnit level:

```
<CSpatialUnits>
    <CSpatialUnit Name="ShakasRapids">
        <DData Name="DPopulation">
                <CurrentValue Value="2.3"> </CurrentValue>
                <OriginalValue Value="2.3"> </OriginalValue>
                <BaseUnit Name = "none"> </BaseUnit>
        </DData>
    <C/SpatialUnit>
<C/SpatialUnits>
```

As processes in LocalComponents level are always run in an *early* time, in this case to specify the tag PprocessTiming as *late* is not necessary. However, we might be interested in know the fish population at a SpatialUnit level (aggregating habitats) or World level (aggregating spatial units), for which *late* time is used. In this example, spatial units contains only 1 habitat, so fish population at SpatialUnit level will be the same as at Habitat level, but we the computed fish population in all or some spatial units could be also aggregated. For example, we could aggregate (so *PAddValue* PSubProcess is used) fish population in the spatial units ShakaRapist and Petronella (we could add all the spatial units that we wish), adding the next XML structure in the ComponentDetailList.xml file at World level.

```
<ComponentDetailList>
<CWorld>
        <PProcess Name = "PFishPopIncreaseSum"
                PProcessType = "PProcess"  PProcessTiming = "Late" >
                <PSubProcess Name = "PCalculateFishPopulationMean"
                        PProcessType = "PAddValue" >
                        <Input SpatialUnitName="ShakasRapids"
                                HabitatName = "Default"
                                LocalComponentName="CFish"
                                DData="DPopulation"
                                DataType="CurrentValue">
                        </Input>
                        <Input SpatialUnitName="PetronellaReach"
                                HabitatName = "Default"
                                LocalComponentName="CFish"
                                DData="DPopulation"
                                DataType="CurrentValue">
                        </Input>
                        <Output SpatialLinkName="GLOBAL"
                                DData="DFishPopulationSum"
                                DataType="CurrentValue">
                        </Output>
                </PSubProcess>
        </PProcess>
 </CWorld>
```

Note that LocalComponent, Habitat and SpatialUnit name are described for each DData input used. Moreover, SpatialLink name is described for DData output as "GLOBAL", according to the World level. When a PProcess is in the World level, all DData used in that process should be also described in the QnDWorld.xml file, as it is showed below. Other *late* Pprocesses can be implemented using the PSubProcesses *PMeanValue*, *PSubtractValue* or *PIfEqual*.

```
<CWorld>
<DData Name = "DFishPopulationSum">
    <CurrentValue Value="1.00"> </CurrentValue>
    <OriginalValue Value="1.00"> </OriginalValue>
    <BaseUnit Name = "none"> </BaseUnit>
 </DData>
```

The shown above is just an example of how implement in QnD a process. Different processes can be implemented, showing the relations between different local components. For example, the effect of CB concentration in sediment in people who eat fish can be implemented, computing the DData showed in Figure 9.

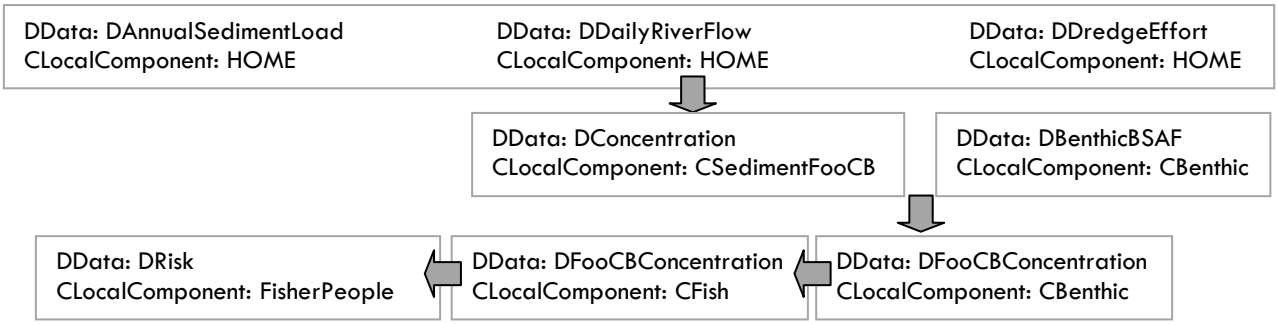


FIGURA 9. ESCHEME OF THE RELATIONS FROM SEDIMENTFOOCB LOCALCOMPONENT TO CFISHERPEOPLE LOCALCOMPONENT

Moreover, in this example, some management decisions are implemented, adding in the DetailList.xml file the DData *DSpatialUnitAverageMeHg, DReleaseDecision, DRestrictionDecision*, common to all the spatial Units.

```
<CSpatialUnit Name="ALL" >
        <DData Name = "DSpatialUnitAverageMeHg">
                <CurrentValue Value="1.00"> </CurrentValue>
                <OriginalValue Value="1.00"> </OriginalValue>
                <BaseUnit Name = "ppb"> </BaseUnit>
        </DData>
        <DData Name = "DReleaseDecision">
                <CurrentValue Value="0.00"> </CurrentValue>
                <OriginalValue Value="0.00"> </OriginalValue>
                <BaseUnit Name = "none"> </BaseUnit>
        </DData>
        <DData Name = "DRestrictionDecision">
                <CurrentValue Value="0.00"> </CurrentValue>
                <OriginalValue Value="0.00"> </OriginalValue>
                <BaseUnit Name = "none"> </BaseUnit>
        </DData>
</CSpatialUnit>
```

Preparation of the Management.xml file

Characteristics regarding the simulation time should be defined here, such as the start and end dates and the step time (Table 9). Different time step (short and long time) can also be defined.

TABLA 9. CLOCALCOMPONENT AND DDATA DESCRIBED IN EACH CLOCALCOMPONENT LEVEL

=Name	EndYear	EndMonth	EndDay	EndHour	TimeUnit	Value
DStartDate	1950	1	1	0		1950
DEndDate	1993	1	1	0		1
DshortTimeStep					day	1
DLongTimeStep					day	365

An example of this structure in XML schema would be:

```
<ManagementList>
        <DStartDate>
                <StartYear>1950</StartYear>
                <StartMonth>1</StartMonth>
                <StartDay>1</StartDay>
                <StartHour>0</StartHour>
        </DStartDate>
        <DShortTimeStep>
                <TimeUnit>day</TimeUnit>
                <Value>1</Value>
        </DShortTimeStep>
        *And the same for DEndDate and DLongTimeStep….
```

Moreover, CScenarios are defined. CScenarios are the primary ways to drive the model and record results. In this example, the *Normal Year CScenario* is defined as is showed in Table 10, describing

the DDriverData used. It answers the question "How is the flow?" and "How is the sediment load?". Some characteristics regarding DDriverData, such as base and time units, original, upper and lower limit values, have to be defined, as well as the type of stochastic distribution of the variable.

TABLA 10. DDRIVERDATA OF SIMULATED FLOW CDRIVER

Tag name/Text	=Name	BaseUnit	TimeUnit	Upper Limit	Lower Limit	Original Value	Stochastic Distribution*
DDriverData	Day	None	day	31.00	1.00	1.00	
DDriverData	River Flow	CubicMeters	month	700.00	300.00	300.00	Uniform
DDriverData	Sediment Load						

*UpperLimit and LowerLimit target have to be defined

Besides different Drivers can be defined here, for example, a *RiverFlow2* could be defined and also added in the QnDWorld.xml file in the Spatial Unit level involving.

An example of this structure in XML schema would be:

```
<CScenario Name="Normal Year">
    <CDriver Name = "Simulated River Flow">
        <DDriverData Name = "River Flow">
                <BaseUnit Name = "CubicMeters"> </BaseUnit>
                <TimeUnit Name = "month"> </TimeUnit>
                <UpperLimit Value = "700.00"> </UpperLimit>
                <LowerLimit Value = "300.00"> </LowerLimit>
                <OriginalValue Value = "300.00"> </OriginalValue>
                <StochasticDistribution Name = "Uniform">
                        <UpperLimit Value = "700.00"> </UpperLimit>
                        <LowerLimit Value = "300.00"> </LowerLimit>
                </StochasticDistribution>
        </DDriverData>
        *And the same for another DDriverData….
```

This Section holds the various options for the User Interface, maps and legends, charts, warning light and management sliders have to be described (see Tables 11-14). Moreover, the working directory where the input files are and the title of the project should be defined:

```
<UserInterface>
        <WorkingDir>C:\QnDModel\QnD_v1_1\InputFiles\FooRiver\</WorkingDir>
            <Title>QnD:FooRiver</Title>
```

A "Base Map" can be described, from the shapefile, showing as tooltip the field of the shapefile "QNDXMLNAME", which will be related to the SpatialUnit. Moreover many other fields of the shapefile can be described in order to be showed in the legend, such as FLOW, SEDIMENT, %SHALLOW, %RIFFLE, etc. Some properties regarding variables displayed in the map, such as upper and lower bounds, number of groups, legend name, etc. have to be described here.

Also, other additional layers can be described in order to be displayed in the map, in this example a map of roads is also displayed (Figure 10 and Table 12).

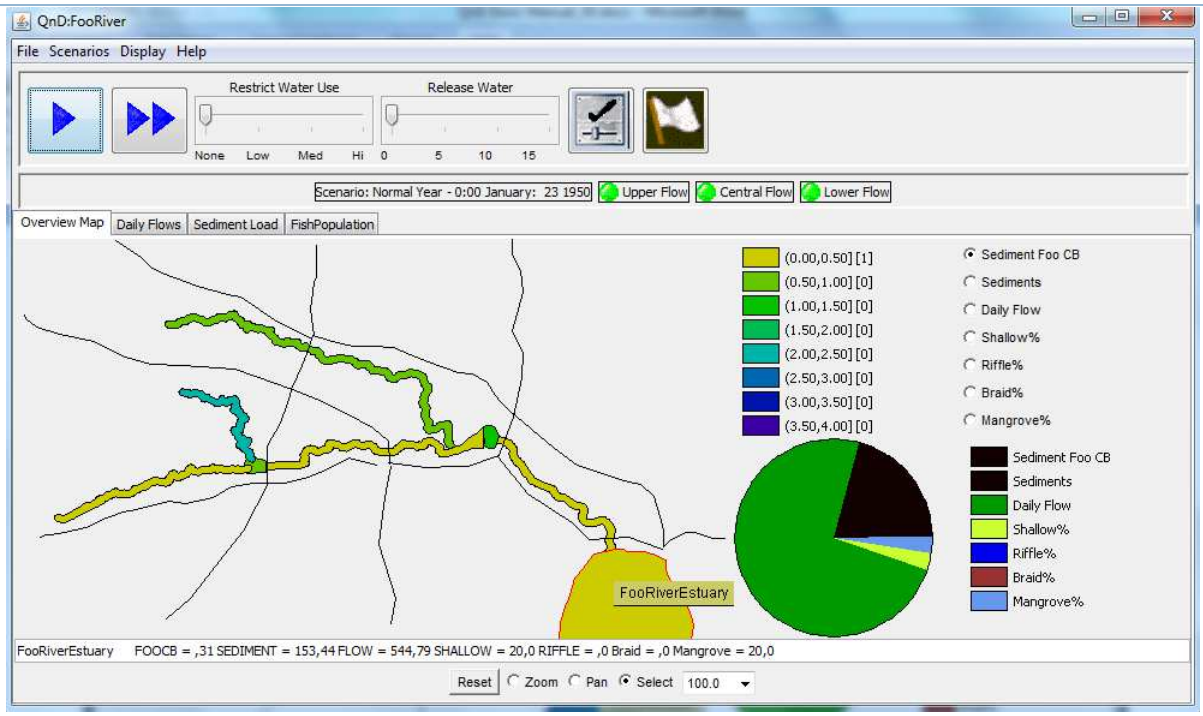


FIGURA 10. INFORMATION OF THE SHAPEFILE

TABLA 11. LAYERS AND LEGENDS FOR MAPS

BasicLayer Name			="Base Map"
Shapefile			c:/QnDModel/QnD_v1_1/InputFiles/FooRiver/fooriver
ToolTip			QNDXMLNAME
	Legends	**LegendName**	="FLOW">
		NiceName	Daily Flow
		NumOfGroups	10
		LowerBound	0.0
		LowerColor	#CCCC00
		UpperBound	1000.0
		UpperColor	#990066
		PieChartColor	#009900
		DData Name*	="DDailyRiverFlow"
Layer Name			="Roads"
	Shapefile		C:/QnDModel/QnD_v1_1/InputFiles/FooRiver/fooriver_fakeroads
	Theme	**Name**	Roads
		IsFilled	False
		IsOutlined	True
		LineColor	#000000
		LineWidth	1
		FillColor	#000000

*DData Type should be defined, eg: "CurrentValue". For SHALLOW, RIFFLE, Braid and Manglove, a CHabitat Name including DData Name have to be described, eg: CHabitat Name="Default", DData Name ="DShallowPercentArea

An example of this structure in XML schema would be:

```
<MAP>
<BasicLayer Name="Base Map">
        <Shapefile>c:/QnDModel/QnD_v1_1/InputFiles/FooRiver/fooriver</Shapefile>
                <ToolTip>QNDXMLNAME</ToolTip>
                        <Legends>
                        <Legend Name="FLOW">
                                <NiceName>Daily Flow</NiceName>
                                <NumOfGroups>10</NumOfGroups>
                                <LowerBound>0.0</LowerBound>
                                <LowerColor>#CCCC00</LowerColor>
                                <UpperBound>1000.0</UpperBound>
                                <UpperColor>#990066</UpperColor>
                                <PieChartColor>#009900</PieChartColor>
                                <DData Name="DDailyRiverFlow"
                                        DataType="CurrentValue">
                                </DData>
                        </Legend>
        *And the same for another Shapefiles, Legends of Themes….
```

DDrivers, input and output DData can be showed in time charts, and also DData in different SpatialUnits can be displayed together. For example, Figure 11 shows the time charts for the DData DDriver *daily flow* in the SpatialUnits Shakarapids and PetronellaReach.

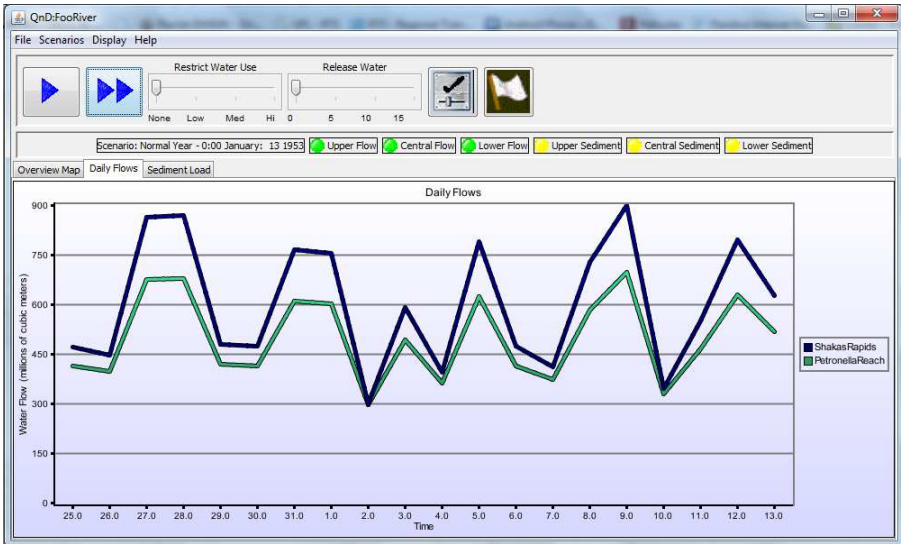


FIGURA 11. TIME CHARTS OF DAILY FLOW IN THE SPATIALUNITS SHAKASRAPIST AND PETRONELLAREACH

TABLA 12. TIMESERIES CHARTS

TimeSeriesChart			="Daily Flows"
	YAxisTitle		Water Flow (millions of cubic meters)
	YAxisLowerBound		0.0
	YAxisUpperBound		2.00
	XAxisTimeStepWidth		10.00
	CSpatialUnit Name		="ShakasRapids"
		DDataName	="DDailyRiverFlow"
		DDataType	="CurrentValue"
	CSpatialUnit Name		="PetronellaReach"
		DDataName	="DDailyRiverFlow"
		DDataType	="CurrentValue"

An example of this structure in XML schema would be:

```
<CHARTS>
        <TimeSeriesChart Name="Daily Flows">
                <YAxisTitle>Water Flow (millions of cubic meters)</YAxisTitle>
                 <YAxisLowerBound>0.0</YAxisLowerBound>
                 <YAxisUpperBound>2.0</YAxisUpperBound>
                <XAxisTimeStepWidth>10.0</XAxisTimeStepWidth>
                <CSpatialUnit Name="ShakasRapids">
                <DData Name="DDailyRiverFlow"
                        DataType="CurrentValue">
                 </DData>
                </CSpatialUnit>
                <CSpatialUnit Name="PetronellaReach">
                <DData Name="DDailyRiverFlow"
                        DataType="CurrentValue">
                 </DData>
                </CSpatialUnit>
        </TimeSeriesChart>
```

Figure 12 shows time chart for the output DData *DPopulation*. In this case DData id referred to a certain LocalComponent, CFish, and to a Habitat, so this has to be defined in the XML structure.

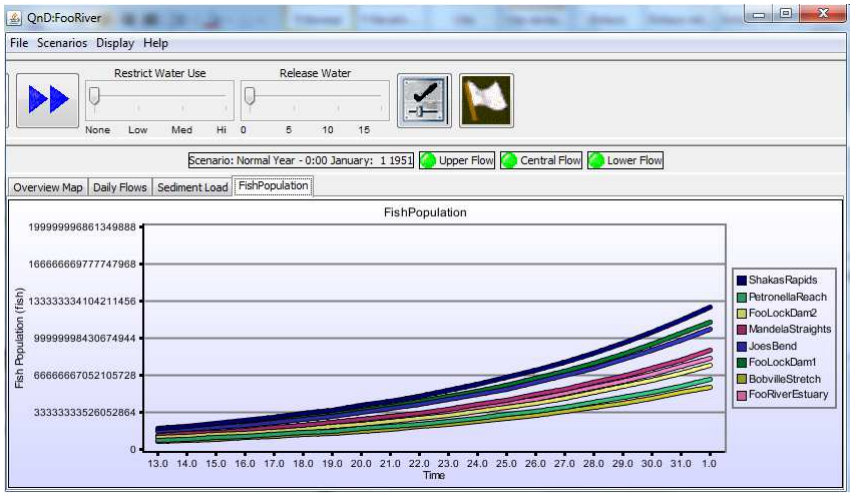


FIGURA 12. TIME CHARTS OF DPOPULATION IN THE CFISH LOCALCOMPONENT IN THE 8 SPATIALUNITS

```
<TimeSeriesChart Name="FishPopulation">
        <YAxisTitle>Fish Population (fish)</YAxisTitle>
        <YAxisLowerBound>0.0</YAxisLowerBound>
        <YAxisUpperBound>1000.0</YAxisUpperBound>
        <XAxisTimeStepWidth>20.0</XAxisTimeStepWidth>
        <CSpatialUnit Name="ShakasRapids">
                <CHabitat Name = "Default">
                        <CLocalComponent Name="CFish">
                                <DData Name="DPopulation"
                                        DataType="CurrentValue">
                                </DData>
                        </CLocalComponent >
                </CHabitat>
        </CSpatialUnit>
</TimeSeriesChart>
```

Other option of QnD GUI is to show some warning lights according DData that could have temporal change and would be interesting to know the exceeding of a certain threshold (Figure 13 and Table 13). Different thresholds can be defined for different states in green, yellow and red color, in order to show for example the risk level in FisherPeople in a SaptialUnit in the upper, central and lower catchment.

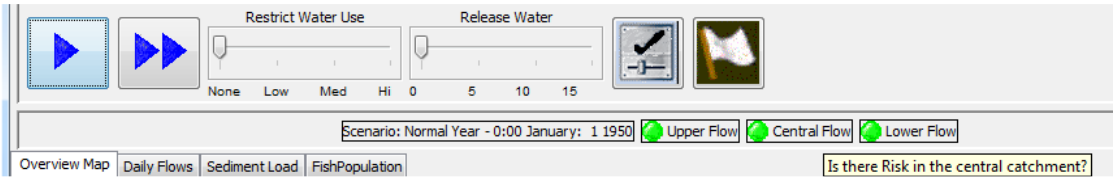


FIGURA 13. WARNING LIGTHS FOR RISK LEVEL IN FISHER PEOPLE

TABLA 13. WARNING LIGHT

Light Name			="Upper Flow">
	ToolTip		Is there risk in the upper catchment?
	GreenThreshold		0.01
	YellowThreshold		0.10
	RedThreshold		1.00
	CSpatialUnit Name		="ShakasRapids">
		CHabitat Name	Default
		CLocalComponent Name	CFisherPeople
		DData Name	="DRisk"
		DataType="	="CurrentValue">

An example of this structure in XML schema would be:

```
<WarningLights>
        <Light Name="Upper Flow">
        <ToolTip>Is there Risk in the lower catchment?</ToolTip>
        <GreenThreshold>0.01</GreenThreshold>
        <YellowThreshold>0.10</YellowThreshold>
        <RedThreshold>1.00</RedThreshold>
        <CSpatialUnit Name="ShakasRapids">
                <CHabitat Name = "Default">
                        <CLocalComponent Name="CFisherPeople">
                                <DData Name="DRisk"
                                        DataType="CurrentValue">
                                </DData>
                        </CLocalComponent >
                </CHabitat>
        </CSpatialUnit>
        </Light>
*And the same for other Lights….
```

Finally management options, described in the DetailList.xml file can be implemented by sliders (Figure 12 and Table 14).

TABLA 14. MANAGEMENTSLIDER

ManagementSlider	**Caption**		Restrict Water Use
	OptTitle		None Low Med Hi
	ToolTip		Impose Release Restrictions? none/low/med/high?
	Minimum		Minimum
	Maximum		3
	InitialValue		0
	DData Name		=“DRestrictionDecision”
		DDataType	=“CurrentValue”

An example of this structure in the XML schema file would be:

```
<ManagementSlider>
        <Caption>Restrict Water Use</Caption>
        <OptTitle>None    Low    Med    Hi </OptTitle>
        <ToolTip>Impose Release Restrictions? none/low/med/high?</ToolTip>
        <Minimum>0</Minimum>
        <Maximum>3</Maximum>
        <InitialValue>0</InitialValue>
        <DData Name="DRestrictionDecision"
                DataType="CurrentValue">
        </DData>
</ManagementSlider>
*And the same for other ManagementSliders….
```

Preparation of the Output.xml file

In this file any DData can be included. Output name should be defined as well as the higher levels where the DData are included (SpatialUnit, CHabitat and CLocalComponent). Moreover, a name has to be assigned to the .csv file where output results are going to be written.

TABLA 15. TIMESERIES OUTPUT

Target					Example
File Name					="QnD_FooRiver_Fish.csv"
	CSpatialUnit Name				="ShakasRapids"
		CHabitat Name			= "Default"
			CLocalComponent Name		="CFish"
				DDataName	="DPopulation"
File Name					="Mean Values"
File Name					="Standard Deviations"

An example of this structure in the XML schema file would be:

```
<OutputList>
        <TimeSeriesOutputFiles>
        <File Name="QnD_FooRiver_Fish.csv">
                <CSpatialUnit Name="ShakasRapids">
                        <CHabitat Name = "Default">
                                <CLocalComponent Name="CFish">
                                        <DData Name="DAbundance"
                                                DataType="CurrentValue">
                                        </DData>
                                        <DData Name="DFooCBConcentration"
                                                DataType="CurrentValue">
                                        </DData>
                                </CLocalComponent>
                        </CHabitat>
                </CSpatialUnit>
        </File>
        *And the same for another TimeSeriesOutputFiles that could be defined….
```

REFERENCES

[1] Kiker, G.A., Thummalapalli, R. 2009. How2QnD: Design and construction of a game-style, environmental simulation engine and interface using UML, XML and Java. Advances in Modeling Agricultural Systems, 103-119.