Supplementary Material: ODD Protocol for Aqua.MORE 1.0

Software availability

Name of software: Aqua.MORE 1.0 – **A**gent-based **MO**deling of **RE**sources (here: of the resource water)

Developer: Nico Bahro and Lisa Huber

Contact details: Lisa Huber, Department of Ecology, University of Innsbruck, 6020 Innsbruck, Austria, lisa.huber@uibk.ac.at, T: +43-512-507-51627

Availability: Free download at the CoMSES Network
https://www.comses.net/codebase-release/febcb4a9-1af7-447c-a4da-78ba6bbec86a/

Software required: NetLogo 6.0.4 or higher, programming language: NetLogo

1. Purpose

Aqua.MORE (**A**gent-based **MO**deling of **RE**sources, here: of the resource water) is an agent based modelling (ABM) approach to simulate the resource flow and social interaction in a coupled natural and social system of water supply and demand. Model version 1.0 is configured to represent an idealized case study site in the Alps. In this synthetic valley, there are two mountain creeks passing one village each (village 1 and village 2), which are then merged to one river, passing village 3. Due to varying economic orientations of the villages (e.g. touristic, agricultural), they differ in their main water demand sectors. The aim of the model is the simulation of scenarios and the interpretation of their outputs, (1) to assess the effects of a hydropower plant in village 2 and (2) to compare different behavioral strategies of the irrigation manager in village 3.

2. Entities, state variables and scales

The NetLogo model environment consists of 15 x 17 patches. One *tick* represents one hour.

The higher-level entities of the agents are *waters*, *managers* and *users. Waters* represent discrete quantities of the resource water; *users* consume the resource, while the *managers* regulate and manage resource flows to *users*. To represent the peculiarities of the idealized case study site, the following lower-level entities (breeds) were realized (Table 1).

Table 1. Lower-level entities (breeds) in village 1, 2 and 3.

Higher-level entities	Lower-level entities (breeds)	village 1 (n)	village 2 (n)	village 3 (n)
Waters	-	-	-	-
Managers	*irrigation managers*	0	0	1
Users	*farmers*	0	0	6
	hydropower users	0	1	0
	snowmaking reservoirs	1	0	0
	inhabitants	1	1	1
	hotels	1	0	0

Waters have the obligatory state variable 'amount_water' representing the amount of water available at this point. *Managers* and *users* must have the state variables 'residualwater', 'demand', 'scarcity' and 'excess'. Additional individual state variables were attributed to several breeds (Table 2).

Table 2. State variables of the various lower-level entities (breeds) of *users* and *managers* in the Aqua.MORE application. Obligatory variables are marked with *.

Variable type	Name of the state variable in the code	Farmers	Hotels	Hydropower users	Inhabitants	Irrigation managers	Snowmaking reservoirs	Variable description
Demand + auxiliary variables	residualwater*	x	x	x	x	x	x	residual water the *manager/user* is obliged to leave in the stream
	demand*	x	x	x	x	x	x	amount of water the *manager/user* wants to extract in the current tick
	concession					x		legal water concession for irrigation
	irrigated-area	x						units of irrigated area (agricultural fields)
	specdemand	x						specific water demand per unit of irrigated
	initial-demand		x	x			x	initial values for 'demand' (read in via csv
	trend-factor		x				x	factor for the change of the 'demand' in comparison to the 'initial-demand'
	tradewater-total	x						amount of water a *user* gets from other *users* as a result of negotiation; in total
	tradewater-thisyear	x						amount of water a *user* gets from other *users* as a result of negotiation; traded in the current
Monitoring variables	scarcity*	x	x	x	x	x	x	amount of water lacking to satisfy the 'demand' of the *manager/user* in the current
	excess*	x	x	x	x	x	x	amount of water exceeding the 'demand' of the *manager/user* in the current tick
	max-scarcity-10y	x	x		x	x		maximum value of 'scarcity' that had happened within the last 10 years
	max-scarcity-1y	x						maximum value of 'scarcity' that had happened within the last year
	max-rel-scarcity-1y		x				x	maximum relative scarcity (= 'scarcity' / 'demand') within the last year
	min-excess-1y	x						minimum value of 'excess' that had happened within the last year
	max-rel-scarcity-10y	x	x	x	x		x	maximum relative scarcity (= 'scarcity' / 'demand') within the last 10 years

3. Process overview and scheduling

Process overview. *Waters* are created every tick at the upper border of the model environment and are moving downwards all the way through it. *Managers* and *users* are not in motion; they are set at predefined locations along the resource flow. In every *tick*, they extract water according to their 'demand', but can be restricted to leave a certain amount of residual water ('residualwater'). In contrast to the *users*, the *managers* do not consume the resource themselves, but redirect all or a part of it to the associated *users;* i.e. the *irrigation manager* redirects irrigation water to the *farmers.*

Scheduling. Setup is done once at the start of the model. Here, all external data files (file format: comma separated variables, *csv*) are imported and stored as lists, i.e. the two files for water 'inflow' and the 'demand' of the *hotels,* the *snowmaking reservoirs* and the *hydropower users*. Moreover, the

default symbol for each breed in the GUI is defined, the *managers* and *users* are built and placed in the model environment, initial variables are defined and the *tick* counter is reset.

The runtime procedures are run as an infinite loop that can be started and stopped by the model user with the 'Go' button. Within each *tick*, eight runtime procedures are processed (Table 3).

Table 3. Runtime procedures of Aqua.MORE.

Procedure name	What is done?
to create-waters	one *water* is generated at the upper border of the model environment, its variable ‚amount_water' is set according to the 'inflow' list (imported from data file)
to move-waters	all *waters* move one step forward
to update-demands	the variable ‚demand' of the *managers* and the *users* is updated according to the specific submodel
to watermanagers-extract	*watermanagers* extract water (i.e., decrease the ‚amount_water' of *waters* passing by according to their updated variable ‚demand', under consideration of ‚residualwater', resulting in values of ‚scarcity' or ‚excess') and send new *waters* to associated *users*
to waterusers-extract	*users* extract water (i.e., decrease the ‚amount_water' of *waters* passing by, according to their updated variable ‚demand', under consideration of ‚residualwater', resulting in values of ‚scarcity' or ‚excess')
to write-maxandmin	maximum, mean and/or minimum values of ‚scarcity' and ‚excess' of *managers* and *users* are recorded
to measure-runoff	‚amount_water' of the *water* at the bottom of the model environment is recorded
to kill-waters	all *waters* that have reached the bottom of the model environment or that are exhausted (variable 'amount_water' = 0), are deleted

4. Design concepts

4.1. Basic principles

All *users* and *managers* have a variable 'demand' that represents the agent's demand for water at the current time step. According to their demand, the *users* and *managers* use water, i.e. they decrease the 'amount_water' of *waters* passing by, which can result in situations of water 'excess' or water 'scarcity'. As the behavior and its consequences of the households is not subject of interest in this model simulation, the *inhabitants*' 'demand' is fixed for the whole model run. But all further *users* and *managers* regularly adapt their 'demand' within the procedure *to update-demands* (Table 3) based on predefined trends and/or decision rules.

4.2. Emergence

The variables 'amount_water' of *waters* and the 'demand' of *managers* and *users* are adaptive traits and the basis for the calculation of excess or scarcity situations, which are stored as 'scarcity' and 'excess' in short-term or as 'max-scarcity-10y', 'max-scarcity-1y', 'min-excess-1y', 'max-rel-scarcity-1y' or 'max-rel-scarcity-10y' in the long-term (Table 2). Therefore, we expect the monitoring variables to vary in complex ways when particular characteristics or behavior of *waters*, *managers* and / or *users* change; exploring this unexpected and complex system behavior is the focus of Aqua.MORE.

4.3. Objectives

In Aqua.MORE, the objective of all *managers* and *users* is the prevention or avoidance of scarcity situations - for themselves and/or for other agents. More precisely, *managers* and *users* want the

variable 'scarcity' to be as low as possible, or - in long term - the variables 'max-scarcity-1y' and 'max-scarcity-10y' (or 'max-rel-scarcity-1y' / 'max-rel-scarcity-10y') to be as low as possible.

4.4. Adaptation

Rules for the 'demand' adaptations are empirical/heuristic rules and assumption-based rules [1]; for the detailed decision rules see chapter 6 *Submodels*. 'Demand' is either adapted according to predefined trends (e.g. due to climatic or economic changes) or is subject to decision processes. Decision processes are triggered and controlled by scarcity experiences in the previous simulation years. The agent behavior therefore carries characteristics of both the deliberative and the reactive approach, according to the agent classification of Bandini et al. [2]. As an example, the *irrigation manager* checks scarcity experiences of the associated *farmers* as well as of the *inhabitants* in the same village within the previous ten years to decide whether to increase or to decrease the amount of irrigation water.

4.5. Learning and Prediction

In the presented idealized case study, the agents neither change their behavioral rules as consequence of their experience nor do they estimate future consequences of their decisions. These features, however, could be included in further case studies.

4.6. Sensing

In most of the submodels of the idealized case study (see chapter *6 Submodels*), *users* or *managers* explicitly know about their own scarcity or scarcity of other agents and consider this in their water use decisions. E.g., the *irrigation manager* knows if the *inhabitants* and *farmers* had experienced any scarcity situations in the previous ten years.

4.7. Interaction

The main interaction in Aqua.MORE is the use of water, whereby *managers* or *users* interact directly with *waters*. Moreover, several case study specific interactions are defined; for details see chapter 6 *Submodels.*

4.8. Stochasticity

Several submodels are driven by randomly created numbers (Table 5). Stochasticity is used here to cover a wide variety of external trends that are not exactly predictable, e.g. the magnitude of rising tourism in the future.

4.9. Observation

In every *tick,* 'scarcity' and 'excess' are calculated for every *manager* and every *user* in the procedures *to watermanagers-extract* and *to waterusers-extract*. Based on these short-term monitoring variables, further long-term expressions for scarcity or excess are monitored, i.e. 'max-scarcity-10y', 'max-scarcity-1y', 'min-excess-1y', 'max-rel-scaricty-1y', 'max-rel-scarcity-10y'; see Table 2. Additionally, the 'discharge' is measured, i.e. the 'amount_water' of the *water* at the lower border of the model environment is monitored by the procedure *to measure-runoff*.

5. Input and initialization

For the first year of model run, the variables need to be initialized.

The initial natural available water on hourly basis is read in from two external csv-formatted data files ('inflow1.csv' for village 1 and 'inflow2.csv' for village 2). We assume the inflow being dominated by

the melt of snow and glacier ice in their streamflow regime, having lowest inflow in January and highest inflow in July (Figure 1a).

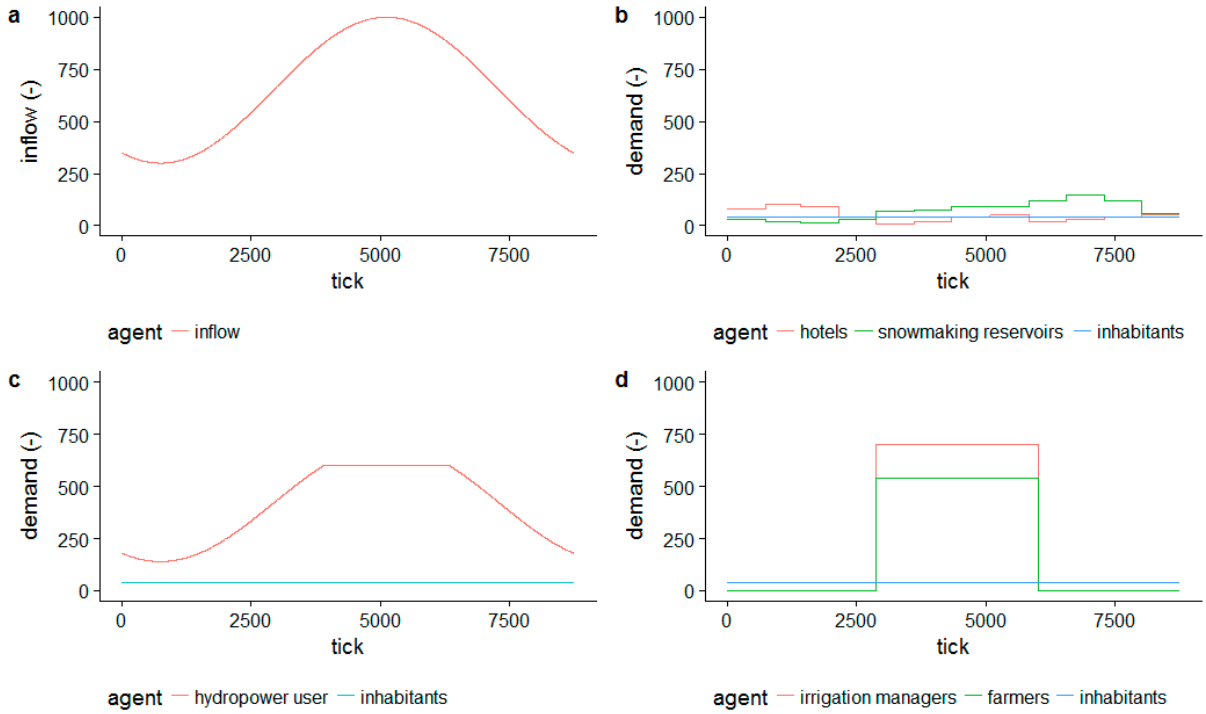

Figure 1. Initial values for the first year of model run (=8760 ticks) of (a) inflow in village 1 and village 2, (b) demands in village 1, (c) demands in village 2, (d) demands in village 3 as defined in the imported data files or directly in the code.

The initial values for the 'demand' of *managers* and *users* are either defined in the code (Table 4) or imported via data files (recommended file format: csv):

- For the *snowmaking reservoirs* and *hotels* in village 1, 'initial-demands' for the first year are imported from data files ('hotel_demand.csv' and 'snowmaking_demand.csv'). Annual changes are realized with 'trend-factor', which is initialized and changed regularly in the code.
- For the *hydropower user* in village 2, 'initial-demands' are imported from a data file ('hydropower_demand.csv').
- For the *irrigation manager* in village 3, the 'demand' is calculated from the auxiliary variable 'concession', whereby the 'concession' is initialized and adapted regularly in the code (Table 4).
- For the *farmers* in village 3, the 'demand' is calculated from the auxiliary variables 'irrigated-area', 'specdemand' and 'tradewater-total', whereby the additional variable 'tradewater-thisyear' is needed in the code for a regular update of the 'tradewater-total'. All listed variables are initialized and updated directly in the code (Table 4).
- For the *inhabitants* in all three village 1,2 and 3 the 'demand' is initialized directly in the code; for simplification of the case study it is the same value for all of them (Table 4).

Moreover, also the ‚residualwater' is defined directly in the model code for all *managers*/*users*. As the model is kept very simple, ‚residualwater' for all *managers* and *users* (except *farmers*) is 100.

Table 4. Initial values of state variables in the code for the Aqua.MORE application in the Matsch Valley.

Village	Agent	State variable	Initial value	Frequency of change

				(a)
Village 1	**hotels**	trend-factor	1	1
		residualwater	100	-
	inhabitants	demand	40	-
		residualwater	100	-
	snowmakers	trend-factor	1	5
		residualwater	100	-
Village 2	**hydropower user**	residualwater	100	-
	inhabitants	demand	40	-
		residualwater	100	-
Village 3	**irrigation**	concession	700	10
		residualwater	100	-
	farmers	irrigated-area	120	5
		specdemand	4.5	-
		tradewater-total	0	1
		tradewater-thisyear	0	1
		residualwater	0	-
	inhabitants	demand	40	-
		residualwater	100	-

6. Submodels

The main submodel describes the abstraction of water by *managers* or *users* and thereby is the core process of Aqua.MORE. The codes for the procedures *to waterusers-extract* and *to watermanagers-extract* (Table 3) are to a large extent the same. At first, a local variable 'available_water' is calculated from the 'amount_water' of the *water* that is present at the same patch as the *user* / *manager* minus the 'residualwater' the *user* / *manager* is obliged to leave in the stream. The 'available_water' therefore represents the amount of water that can be used in the current time step. The *user* / *manager* extracts water, i.e. the own variable 'amount_water' of the *waters* passing by is decreased according to the 'demand'. Depending on whether the magnitude of 'available_water' is higher or lower than his 'demand', 'excess' or 'scarcity' is recorded, respectively. In the GUI, *users* (usually yellow) and *managers* (usually green) which are experiencing scarcity appear in red. *Watermanagers-extract* only differs from *waterusers-extract* by the *managers* creating new *waters* with a corresponding 'amount_water' (i.e., optimally their 'demand', but no more than all the 'available_water') and redirecting them to the associated *users.*

All further submodels cover the procedures for updating and calculating the 'demands' of the agents (*to update-demands*; Table 3), and are case study specific.

Table 5. Case-study specific procedures for the Aqua.MORE application in the idealized case study site. For explanation of the variables see Table 2. Global variables which are relevant for the scenario simulation are highlighted in dark grey; random processes are highlighted in light grey.

Agent	Variable that is calculated	Frequency	Submodels
hotels in village 1	'trend-factor'	every year	If the *snowmaking reservoirs* and *hotels* in village 1 have 'max-rel-scarcity-1y' < 0.1, • then 'trend-factor' of the *hotel* is increased by 'trend-factor' multiplied by a random number between 0 and 0.05, • else 'trend-factor' of the *hotel* is decreased by 'trend-factor' multiplied by a random number between 0 and 0.10
	'demand'	every tick	'demand' of the *hotels* = 'initial-demands' * 'trend-factor'
snowmaking reservoirs in village 1	'trend-factor'	every 5 years	• If 'trend-factor' of *hotels* > 1.0, then 'trend-factor' of the *snowmaking reservoirs* = 'trend-factor' of the *hotels* • If 'trend-factor' of *hotels* < 0.5, then 'trend-factor' of the *snowmaking reservoirs* = 0.5
	'demand'	every tick	'demand' of the *snowmaking reservoirs* = 'initial-demands' * 'trend-factor'
hydropower user in village 2	'demand'	every tick	if 'yearcounter' >= the global 'startyear-hydropoweruser', • then 'demand' = 'initial-demand', • else 'demand' = 0
irrigation manager in village 3	'concession'	every 10 years	• If any of the n=6 *farmers* has 'max-rel-scarcity-10y' > 0.1 and the *inhabitants* of village 3 have'max-scarcity-10y' = 0, then the *irrigation manager* increases 'concession' by the magnitude of: maximum value of 'max-scarcity-10y' of all 6 *farmers*, multiplied by a global 'safety-coefficient'. • If the *inhabitants* of village 3 have 'max-scarcity-10y' > 0, then the *irrigation manager* decreases 'concession' by the magnitude of 1.1 * ('max-scarcity-10y' of *irrigation manager* + 'max-scarcity-10y' of *inhabitants* of village 3)
	'demand'	every tick	If 'daycounter' between 120 and 250, • then 'demand' = 'concession', • else 'demand' = 0
farmers in village 3	'tradewater-total'	every year	One after the other, every *farmer* with 'max-scarcity-1y' > 0 can loop through all *farmers* with 'min-excess-1y' > 0 and 'tradewater-thisyear' < 'min-excess-1y': 'decision' = random binary number If 'decision' = 1, then 'tradewater-thisyear' of the *farmer* with scarcity is decreased and 'tradewater-thisyear' of the *farmer* with excess is increased for the amount: 'min-excess-1y ' of the *farmer* with excess. At the end of this procedure, 'tradewater-total' of every *farmer* is increased by 'tradewater-thisyear'; afterwards 'tradewater-thisyear' is set back to 0.
	'area'	every 5 years	For every *farmer*, 'area' is increased by a random number between 0 and 15.
	'demand'	every tick	If 'daycounter' between 120 and 250, • then the *farmers* have 'demand', one after the other for 24 hours: 'demand' = 'area' * 'specdemand' + 'tradewater-total', • else 'demand' = 0

7.2. Hotels in village 1

During years of sufficient water supply (i.e. none of the *hotels* or *snowmaking-reservoirs* in village 1 has experienced at least one tick, when available water was satisfying less than 90% of their 'demand'), tourism increases. Hence, we assume the 'demand' of *hotels* to increase every year by 0.0 - 5.0%. If there has been any severe scarcity situation of *hotels* or *snowmaking-reservoirs* in the previous year (i.e. available water is satisfying less than 90% of their 'demand', which corresponds to a 'max-rel-scarcity-1y' > 0.1), the tourist offer could eventually be affected; therefore we assume decreasing 'demand' in this case (0.0 - 5.0%). In contrast to other submodels, the change in tourism can not be seen as a decision of the respective *hotels,* but the result of an external (not simulated) process of tourists deciding for their holiday destination.

7.3. Snowmaking reservoirs in village 1

The 'demand' of the *snowmaking reservoirs* in village 1 is also dependent on tourism and therefore coupled to the 'demand' of the *hotels*. *Snowmaking reservoirs*, however, only adapt every 5 years. During years of sufficient water supply, the 'demand' of the *snowmaking reservoirs* increases to the same extent as the 'demand' of the *hotels*. However, if 'demand' of the *hotels* decreased to 50% or less in comparison to the first year of a model run (i.e., 'trend-factor' of hotels <= 0.5), the slump in tourism leads to financial difficulties and the *snowmaking reservoirs* are cut back to 50% of the 'initial-demand'.

7.4. Hydropower users in village 2

In simulations without hydropower use, the 'demand' of the *hydropower user* is set zero. In simulations with hydropower use, the 'demand' is specified by the imported data from the csv file ('initial-demand').

7.5. Irrigation manager in village 3

The 'demand' of the *irrigation manager* is calculated from the 'concession' only from days 120 to 250, representing the irrigation period in the vegetation growing season. We assume the *irrigation manager* to be bound to a water concession agreement, which can be (successfully) renegotiated every ten years. His/her goal is to avoid scarcity situations of his associated *users*, i.e. the *farmers*. If any of the *farmers* had experienced a severe scarcity situation (i.e. available water is satisfying less than 90% of their 'demand', which corresponds to a 'max-rel-scarcity-10y' > 0.1) within the last 10 years, the *irrigation manager* increases the 'concession' at least for the maximum scarcity situation. Or, if he/she wants to act with foresight, he/she multiplies the maximum scarcity situations with a global variable 'safety_coefficient' to take precautions for future scarcity situations. Assuming agriculture to have lower priority than households, the *irrigation manager* has to respect scarcity situations of *inhabitants* as well. In the case of scarcity of *inhabitants*, 'concession' cannot be increased, but rather needs to be decreased.

7.6. Farmers in village 3

The demand of the *farmers* in Matsch Valley is determined by the irrigated area ('area'), the specific demand ('specdemand') and the negotiated water supply ('tradewater-total').

- The 'area' of every *farmer* is increased every 5 years, explained by both economic decisions and climatic influences: (1) We generally expect farmers to increase their yield and therefore expand the irrigated area. (2) As the irrigation amount and area of irrigated land is predicted to increase due to global warming and associated drought events and/or heatwaves [3], we can also imagine the farmers in the Matsch Valley expanding their irrigated area to more fields.
- The 'specdemand' of every *farmer* is constant.
- The 'tradewater-total' of every *farmer* can be changed at most every year by interactions (negotiations) between *farmers*. Every *farmer* with water scarcity in the previous year can ask every

farmer with excess in the previous year for water units, whereby the decision for or against the trade is taken randomly (fifty percent chance).

References

1. An, L., Modeling human decisions in coupled human and natural systems: Review of agent-based models. *Ecological Modelling* **2012,** 229, 25-36.
2. Bandini, S.; Manzoni, S.; Vizzari, G., Agent Based Modeling and Simulation: An Informatics Perspective. *Journal of Artificial Societies and Social Simulation* **2009,** 12, (4).
3. Fischer, G.; Tubiello, F. N.; van Velthuizen, H.; Wiberg, D. A., Climate change impacts on irrigation water requirements: Effects of mitigation, 1990-2080. *Technological Forecasting & Social Change* **2007,** 74, 1083-1107.