Technical Documentation of the SoNARe model

1 Model implementation

The overall model documented here consists of the main model named SoNARe and two coupled sub-models. This documentation focuses on the SoNARe model which aims to capture farmer decision making and some of the key social characteristics of the Odra case in an agent-based model. The first sub-model is a simple and abstracted hydro-agricultural model that reflects the main environmental characteristics of the target region. It provides the SoNARe agents with feedback about hydrological dependencies and crop yields under fluctuating climatic conditions in the simulated area. The second sub-model is the Simple Agro-Economics Model (SAEM) which models crop yield and production economics and provides SoNARe agents with feedback about the economic aspects of their farming activities. Both sub-models were developed at Wroclaw University of Technology and are documented in detail elsewhere.

This documentation starts with the core architecture of SoNARe. Based on this core the two versions of the SoNARe model were developed which are documented in separate subsequent sections.

The first, more abstract version is named SoNARe-A and takes a quantitative approach to modelling farmers' perceptions and their consideration of social and economic factors that bear on their decisions. Amongst other things, this allows the investigation of sets of scenarios that show shifts from more economically driven farmer populations to more socially driven farmer populations and vice versa.

The second version is more detailed and called SoNARe-D. This version is enhanced in the sense that it includes more data from the actual case study (e.g. an elicited network structure and an additional farmer type) and is thus more evidence-driven than the abstract version. In addition, it attempts to more closely capture the symbolic/qualitative nature of agents' decision-making and differentiates a set of social influencing factors.

In the following, the term SoNARe (without –A or –D postfix) refers to common documentation of both models.

Technically, the SoNARe core model is a hybrid model. The modelling infrastructure (scheduling, data logging, visualisation, etc) is provided by the RePast agent programming framework (Recursive Porous Agent Simulation Toolkit, cf. North et al 2006). The cognitive control structure and decision making of farmers and WPI actors may be modelled using production rules implemented in JESS (the Java Expert System Shell; http://herzberg.ca.sandia.gov/jess/), and JESS's reasoning engine. It opens the possibility to provide Repast agents with cognitively plausible capabilities. Furthermore, the RePast portion of the model provides the necessary functionality to interface with the sub-models SHAM and SAEM. Figure 1 illustrates the technical model architecture.

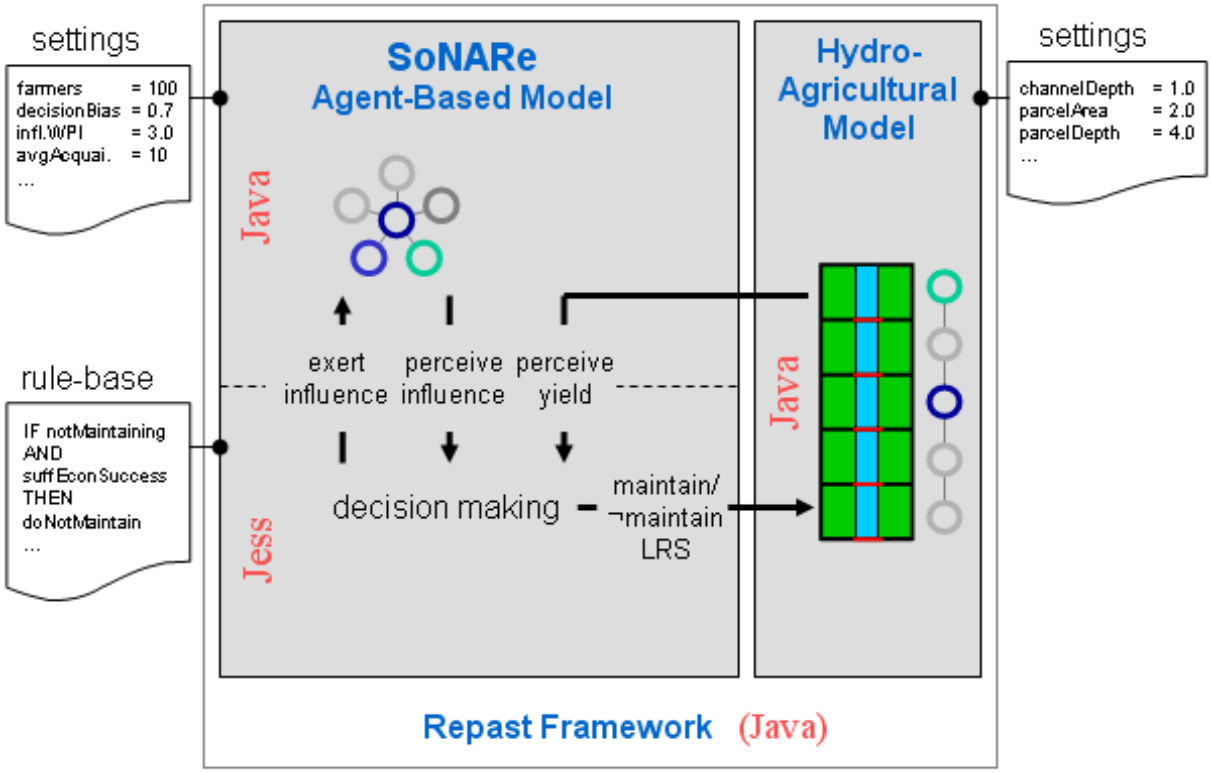

Figure 1: Overview of the integrated model architecture. The integration of Jess and a Jess rule-base is optional. Currently, decision-making is performed procedurally in Java. In the SoNARe-D model the basic architecture is extended to include the economic model SAEM.

1.1 Class Structure

The following UML class diagrams show the main classes of SoNARe-A/SoNARe-D, their relationships and their relationships to the interfaces of the submodels SHAM and SAEM.

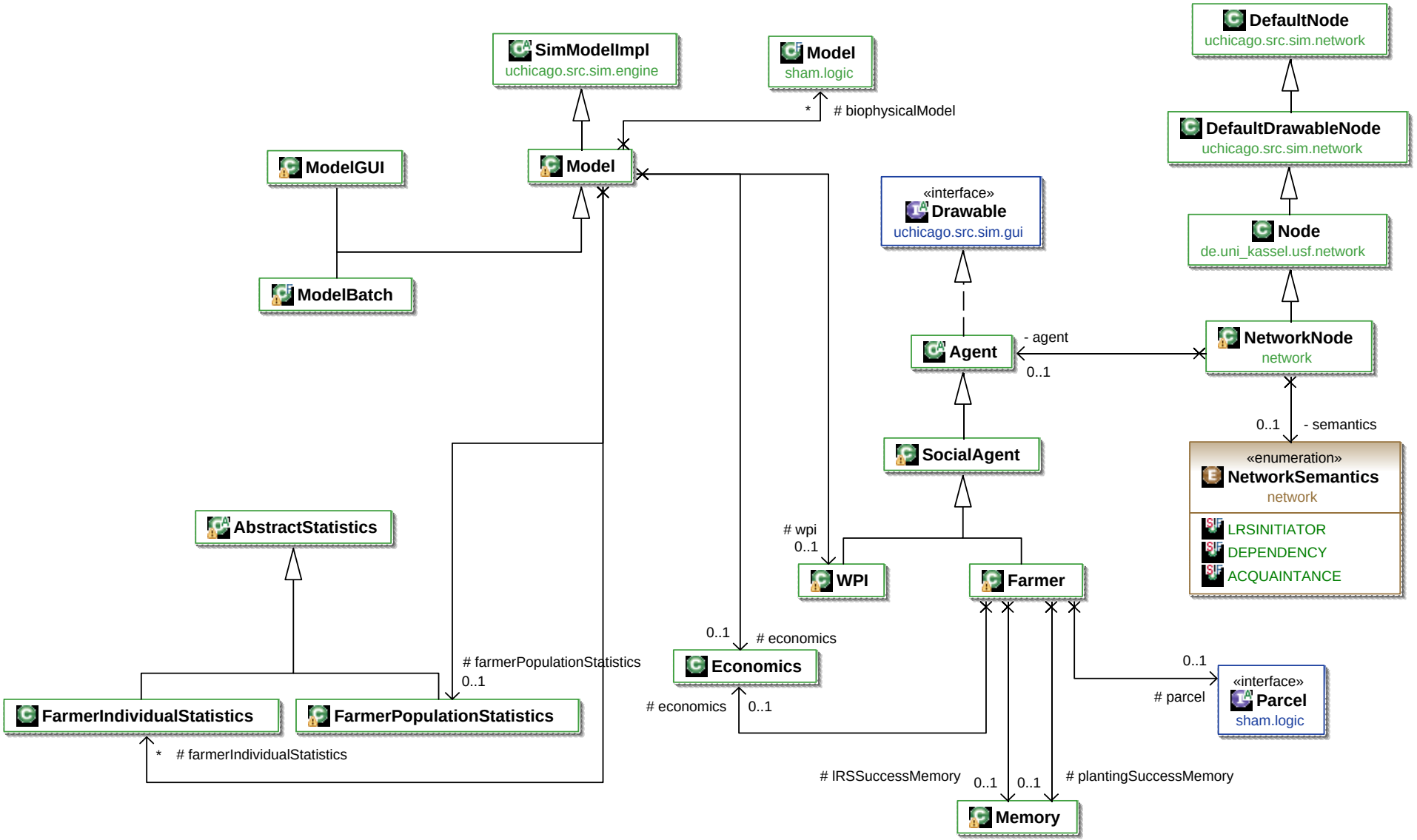

Figure 2: Diagram of the main classes of SoNARe-A and their relationships.

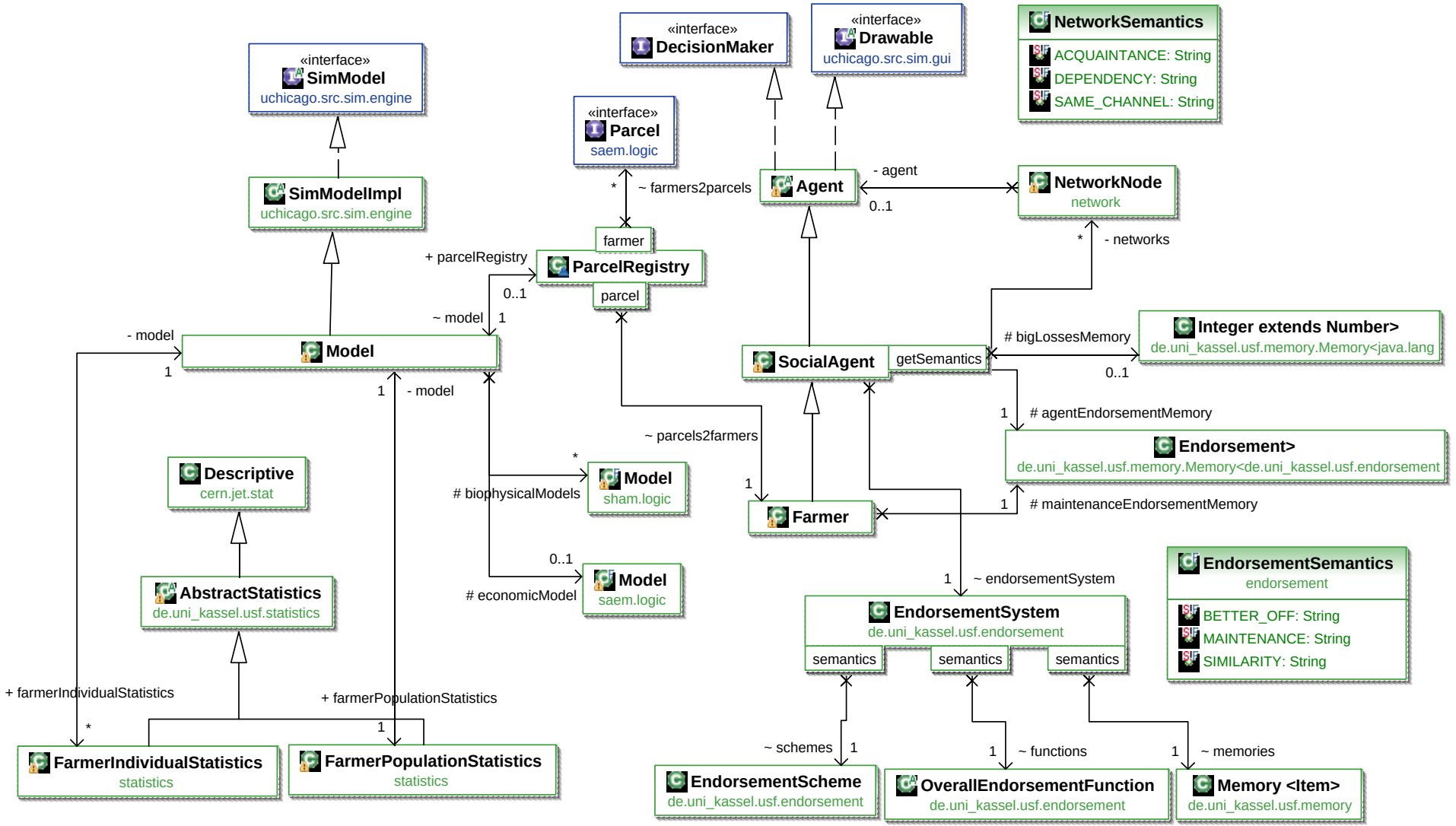

Figure 3: Diagram of the main classes of SoNARe-D and their relationships.

1.2 Required libraries

The SoNARe model requires the following packages:

- colt.jar
- jchart2d-2.1.0.jar
- jess.jar
- jmf.jar
- log4j-1.2.15.jar
- plot.jar
- repast.jar
- sham-0.3.1.jar
- trove.jar
- violinstrings-1.0.2.jar

2 User Guide

2.1 Model Parameters (SoNARe-A)

The following table summarises the parameters of SoNARe-A and their default values.

Environment

`numberOfParcels`	10	• Number of parcels per channel • Parcels have uniform size.
`numberOfChannels`	10	• Number of parallel channels, assumed to be hydrologically independent.
`weatherSequence`	N N W	• Fixed sequence Normal-Normal-Wet
`stoppingTime`	100	• Number of simulation years
`configPath`	sham_scenarios/	• Configuration path of SHAM
`dataLoggingPath`		• File system location where log files are dumped

Agents

`decisionBiasFarmers`		• Sets the common balancing of farmer decision making between economic success and social support perceived.
		• Values above 0.5 stress the economic influence on decision making, values below 0.5 stress the social dimension.
		• Reference settings for the decision scenarios are
		  o Selfish Scenario, set to 1.0
		  o Social Scenario, set to 0.5

`productionCost`	8.0	• Investment in farming activities (excluding LRS) per year • Value was estimated in relation to the maximum attainable yield (market price is 10 units) • Maximum possible profit is 20% (=2 units) of the market price of the maximum yield on a parcel
`costLRS`		• Amount (or work equivalent) per year that has to be invested in the LRS if it is maintained
`profitThresholdFarmers`		• If a farmer's profit in a year drops below `profitThresholdFarmers` then the profit (of that year) is considered "too low" otherwise "ok"
`maxCompensation`	2.5	• Maximum value of compensation that may be paid to an individual farmer
`compensationPolicy`	0	• Compensation Policy
`warmUpYears`	10	• Number of years before compensation policy takes effect

minRetentionTimeFarmers	3	• Farmers memorise whether their past profits were "too low" or "ok" with respect to profitThresholdFarmers. • Memory tokens are memorised with a retention time in years that is fixed per farmer agent • Memory tokens are triples of profit, LRS strategy, and retention time. • E.g. if agent A has a retention of 5 years then a memorised profit will persist for 5 years and after that be removed from the memory. • When evaluating an LRS strategy farmers consult the relevant tokens stored in their memory. • Individual retention times are distributed heterogeneously among agents • retention times are assigned to farmers from a uniform random distribution from [minRetentionTimeFarmers, maxRetentionTimeFarmers] • random seed is set to memRngSeed
maxRetentionTimeFarmers	7	
memRngSeed		
networkType	WS	• Watts-Strogatz network, small world network (ring substrate) generated by the RePast network factory • rewiringProbability=0.1 • connectRadius= avgAcquaintancesDegree / 2 • RandomSeed is the seed of the RNG that is used for the random rewiring
avgAcquaintancesDegree	10	
RandomSeed		
bigLossesThreshold		• The WPI agent has social network links to all farmer agents. • At the end of a simulation year the WPI observes the profit of each farmer and counts those farmers whose profit has dropped below bigLossesThreshold • The WPI keeps a track of these counts over the past 6 years, if the average of this track rises above wpiActivationThreshold then the WPI becomes active and exerts social influence. In all other cases the WPI is passive.
wpiActivationThreshold		
relativeSocialInfluenceWPI		

2.2 Running the model from the Repast GUI (SoNARe-A)

The ModelGUI class of the SoNARe-A model comes with a number of different visualisations. The performance diagrams show yearly values of various model indicators. The performance indicators are defined in the FarmerPopulationStatistics class and in the FarmerIndividualStatistics class. All performance diagrams display data provided by these classes.

As an example, the left hand side of the screen shot below (Figure 4) shows a visualisation of 100 land parcels along ten channels of the LRS (green lines) where red parcels have a neglected LRS, blue ones are maintained. On the right hand side three selected performance diagrams are shown.

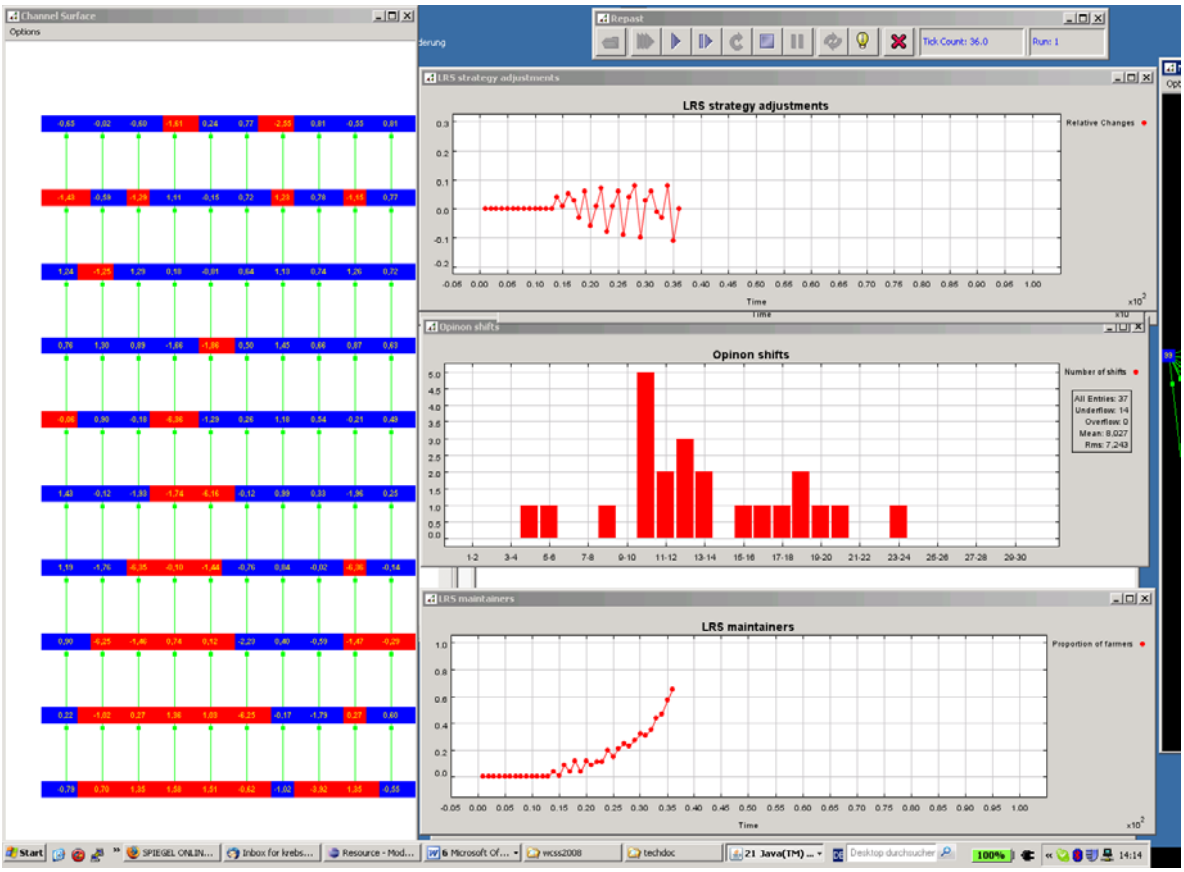

Figure 4: Sample screenshot of some of the visualisations produced by the ModelGUI class of SoNARe-A.

The next screen shot (Figure 5) shows the two network visualisations produced by SoNARe-A. E.g. the window in the foreground shows the acquaintances network where nodes belong to individual farmers. The colour of the nodes corresponds to their respective LRS strategy and the size of the nodes increases with social success.

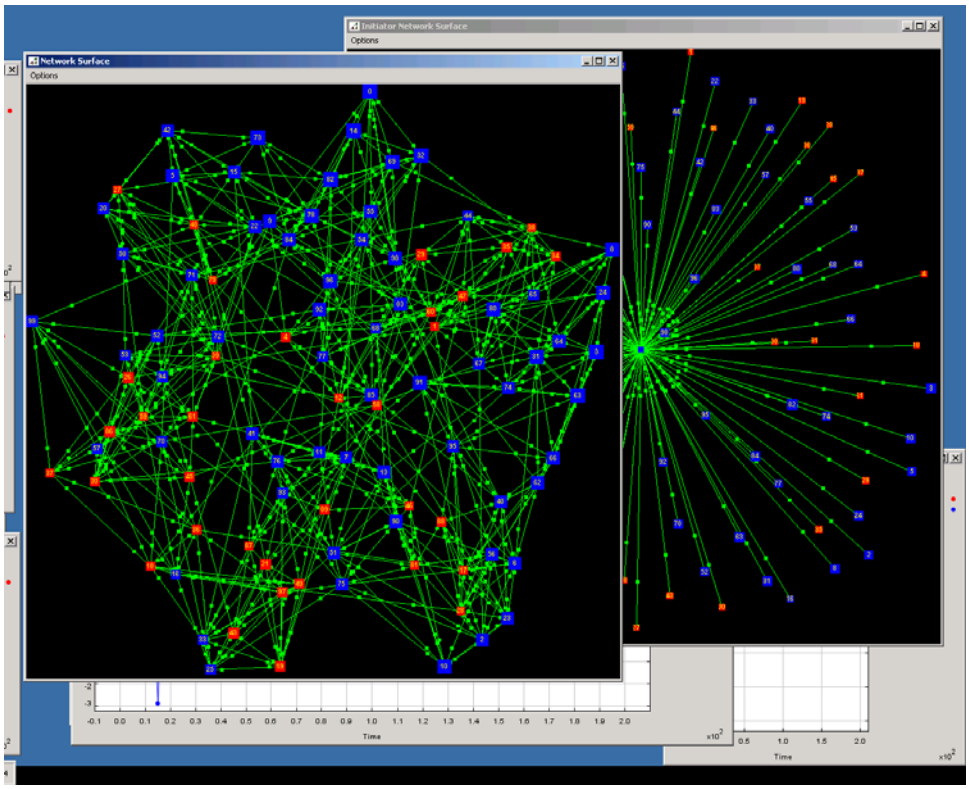

Figure 5: Sample screenshot of some of the visualisations produced by the ModelGUI class of SoNARe-A.

2.3 Running the model in batch mode (SoNARe-A)

The SoNARe-A model may also be run in a silent batch mode which works without the GUI and only writes log files with the simulation results for later offline examination, e.g. to perform sensitivity analyses. This is done by invoking the `ModelBatch` class of SoNARe-A from the command line as follows:

```
java -cp
lib/sonare.jar:lib/repast.jar:lib/jess.jar:lib/colt.jar:lib/trove.jar:lib/viol
instrings-1.0.2.jar:lib/plot.jar:lib/sham-0.2.5.jar
uchicago.src.sim.engine.SimInit -b de.uni_kassel.usf.sonare.ModelBatch
parms/test.pf
```

This assumes that there is a `lib` directory including all required libraries and a model parameter file `test.pf` stored in the `parms` directory. The parameter file may contain any of the model parameters documented above. In addition, ranges for the parameters may be specified — a full description of the parameter file syntax can be found in the RePast documentation.

An excerpt of a sample parameter file that was used for a sensitivity analysis of the SoNARe-A model is given below:

```
runs: 1
MaxCompensation {
	set: 2.5
}
CompensationPolicy {
	set_list: 0 1 2 3 4
}
warmUpYears {
	set: 10
}
```

```
DecisionBiasFarmers {
      set: 0.5
}

...

DataLoggingPath {
  set_string: n:/data/simResults/model_v1/policy/
}
ConfigPath {
  set_string: sham_scenarios/
}
ConfigFile {
      set_string: 2normal1wet.cfg
}
PrintAgentStats {
  set_boolean: false
}
StopAtTick {
      set: 120
}
```

2.4 Model Parameters (SoNARe-D)

Environment

`numberOfParcelsPerChannel`	10	• Parcels have uniform size.
`numberOfBigFarmers`	10	
`numberOfSmallFarmers`	100	
`numberOfFarmerWPIs`	0	• The number of farmers to also act as WPIs. If x is the value set for this variable, then those x farmers will become WPIs who have the x highest number of acquaintances network edges/links. These WPIs will, however, not be connected to all other farmers.
`numberOfNonFarmerWPIs`	1	• The number of additional (non-farmer) agents to act as WPIs. Each non-farmer WPI will be connected to all farmers via the acquaintances network.
`numberWillingAmongstSmallFarmers`	0	• The number of willing small farmers. Willing farmers are assigned a maintenance endorsement scheme that includes the "WPIExertsInfluence" token with a positive endorsement value (see also `willingnessToMaintain`).
`bigFarmerParcelThreshold`	10	• If a farmer agent owns at least as many parcels as this threshold indicates, it is considered to be a big farmer. Currently, all big farmers own exactly this number of parcels. However, an acreage allocation algorithm is implemented in the model which allows to randomly determine the number of parcels individual farmers are assigned.
`farmer2ParcelDistributionMode`	RANDOM_ MAINTAIN_ CONNECTE DNESS	• Determines the mode by which farmers are assigned their parcels. • If set to DEFAULT, the allocation is performed in the order of farmers IDs and each farmer will be assigned the next X number of parcels along the

		current channel (or if X is great, the current and subsequent channels) with X being the total number of parcels the farmer is to own. • If set to RANDOM, allocation is performed at random; a uniform random distribution is used and seeded with `distRngSeed.` • If set to RANDOM_MAINTAIN_CONNECTEDNESS, the next farmer to be assigned its parcels is determined from a uniform random distribution (see above) which is seeded with `distRngSeed` and this farmer will be assigned the next X number of parcels along the current channel (or if X is great, the current and subsequent channels) with X being the total number of parcels the farmer is to own.
`distRngSeed`	1	• see above
`stopAfterYear`	114	• The simulated year after which to stop the simulation.
water level sequence to be fed into the top parcel of a channel	historical water level sequence	• The data represent an average monthly water balance measured in Wroclaw in the period between 1947 and 2003. It was provided as part of SHAM by the project member Grzegorz Holdys, Wroclaw University of Technology.
`dataLoggingPath`		• File system location where log files are dumped.
`shamSaemConfigPath`	config/sham_saem/	• File system path to where the configuration files for SHAM/SAEM are located.
`parmsPath`	config/parms	• File system path to where the (Repast) configuration file for SoNARe is located.

Agents

`networkType`	OdraTrimmed	• The type of network topology to be used for the acquaintances network. OdraTrimmed is the network topology extracted from case study evidence (N=74) which is scaled up or down depending on the number of small farmers.
`randomRewiringProportion`	0.0	• The proportion of edges to be randomly rewired for each node that is newly generated by the scaling algorithm applied to the OdraTrimmed network.
`lossToleranceFraction`	0.9	• If a farmer's current profit falls below this fraction of the mean profit using the current maintenance strategy, it will regard it as a big loss.
`wpiBigLossesMemoryDefaultRetentionTime`	10	• A WPI gets active and remains active as long as the mean number of the small farmers (i.e. those it is acquainted with) with big losses over the past `wpiBigLossesMemoryDefaultRetentionTime` number of years is greater than `wpiActivationThreshold`

`wpiActivationThreshold`	10	• see above
`mediumYearsMemorised` (profit memory)	10	• Profits are memorised with a retention time in years that is fixed per farmer agent • E.g. if agent A has a retention of 5 years then a memorised profit will persist for 5 years and after that be removed from the memory. • When evaluating an LRS strategy farmers consult the relevant profits stored in their memory. • Individual retention times are distributed heterogeneously among agents • Retention times are assigned to farmers from a normal random distribution from [`mediumYearsMemorised-radiusYearsMemorised, mediumYearsMemorised+radiusYearsMemorised`] • random seed is set to `memRngSeed.`
`radiusYearsMemorised` (profit memory)	0	• see above
`memRngSeed`	1	• see above
`agentEndorsementMemoryCapacity`	-1 (unlimited)	• The default capacity set for each small farmer. • The retention mechanism for the memory that stores an agent's endorsements of other agents in the social network is determined by that memory's capacity and its retention time. • If the capacity is set to a negative value (= unlimited capacity), then an endorsement is stored for the number of years defined by the memory's rentention time. • If the capacity is set to 0 or a positive value, then an endorsement is stored for the number of years defined by the memory's rentention time unless the capacity is reached first. • In case the capacity of the memory is reached, the addition of a new endorsement will result in the removal of another endorsement according to the FIFO rule.
`agentEndorsementMemoryDefaultRetentionTime`	10	• see above
`maintenanceEndorsementMemoryCapacity`	-1 (unlimited)	• Analogous to the above, but applying to the agent's memory for maintenance option endorsements.
`maintenanceEndorsementMemoryDefaultRetentionTime`	10	• see above
`willingnessToMaintain`	1	• Each willing small farmer is assigned this endorsement value for the token "WPIExertsInfluence" in its maintenance endorsement scheme.

2.5 Running model from the Repast GUI (SoNARe-D)

The `ModelGUI` class of the SoNARe-D model comes with a number of different visualisations. The performance diagrams show yearly values of various model indicators. As in the SoNARe-A model version, the performance indicators are defined in the model-specific

`FarmerPopulationStatistics` class and in the `FarmerIndividualStatistics` class. All performance diagrams display data provided by these classes.
The left hand side of the screen shot below (Figure 6) shows (part of) the visualisation of 200 land parcels along twenty channels of the LRS (green lines) where red and blue parcels represent small farmers who have a neglected LRS or a maintained LRS, respectively. The grey parcels indicate big farmers who always maintain their LRS. On the right hand side four representative performance diagrams are shown.
Figure 7, in the bottom right corner features a screenshot of a network visualisation produced by SoNARe-D. It shows the acquaintances network where nodes belong to individual farmers. Again, the colour of the nodes corresponds to their respective LRS strategy/agent type and the size of the nodes increases with the attained individual profit.

Figure 6: Sample screenshot of some of the visualisations produced by the ModelGUI class of SoNARe-D.

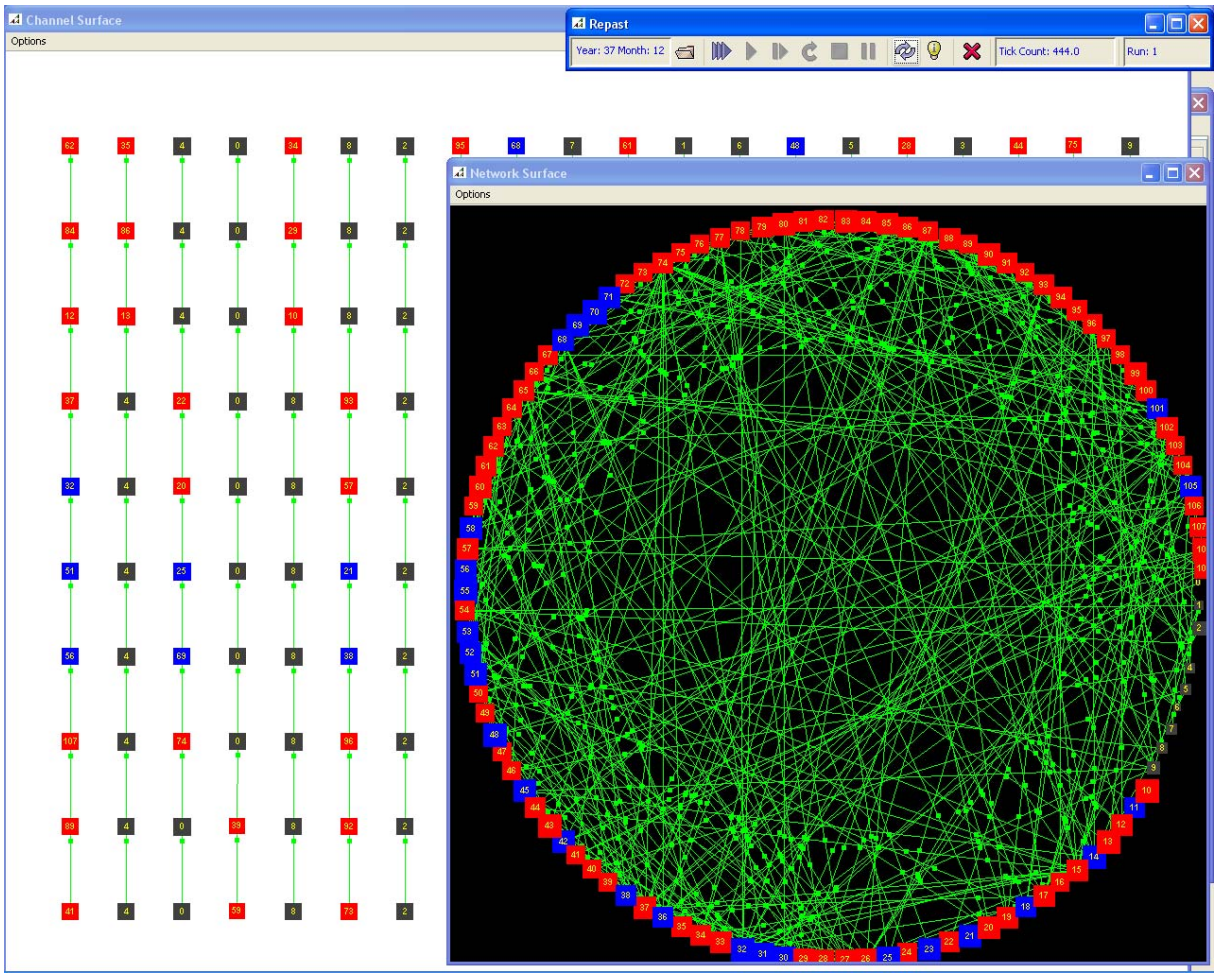

Figure 7: Sample screenshot of some of the visualisations produced by the ModelGUI class of SoNARe-D.

2.6 Running the model in batch mode (SoNARe-D)

SoNARe-D may also be run in a silent batch mode which works without the GUI and only writes log files with the simulation results for later offline examination, e.g. to perform sensitivity analyses. This is done by invoking the `ModelBatch` class of SoNARe-D from the command line as follows:

```
java -cp
lib/sonare5.jar:lib/sham0.3.1.jar:lib/repast.jar:lib/jess.jar:lib/colt.jar:li
b/trove.jar:lib/violinstrings1.0.2.jar:lib/plot.jar:lib/beanbowl.jar:lib/log4
j-1.2.15.jar
uchicago.src.sim.engine.SimInit -b de.uni_kassel.usf.sonare.ModelBatch
config/parms/test.pf
```

This assumes that there is a `lib` directory including all required libraries and a model parameter file `test.pf` stored in the `config/parms` directory. The parameter file may contain any of the model parameters documented above. In addition, ranges for the parameters may be specified — a full description of the parameter file syntax can be found in the RePast documentation.

An excerpt of a sample parameter file that was used for a sensitivity analysis of the SoNARe-D model is given below:

```
runs: 1
```

```
NumberWillingAmongstSmallFarmers {
       start: 0
       end: 100
       incr: 25
       {
       runs: 1
       DistRngSeed {
              start: 1
              end: 10
              incr: 1
              {
              runs: 1
              RngSeed {
                     start: 1
                     end: 3
                     incr: 1
              }
              }
       }
       }
}
StopAfterYear {
  set: 114
}
NumberOfBigFarmers {
  set: 10
}
NumberOfSmallFarmers {
  set: 100
}
BigFarmerParcelThreshold {
  set: 10
}

...

LossToleranceFraction {
  set: 0.9
}
...

DataLoggingPath {
  set_string: n:/data/simResults/model_v2/endorsementSchemes/
}
```

2.7 Model Output

As for the visualisations in the `ModelGUI` class the `ModelBatch` class dumps all model performance indicators defined in the `FarmerPopulationStatistics` class and in the `FarmerIndividualStatistics` class for each time step in a tabular format to a log file, as indicated below for an example population file.

```
Timestamp: Apr 16, 2008 10:11:16 AM
{The values of the fixed model parameters are printed here}

"run" "tick"      "RngSeed"   "DistRngSeed"     "WpiActivationThreshold" ...
  1    12.0          1           1                 3   ...
    ...
    ...

{The values of the fixed submodel parameters (SHAM/SAEM) are printed here}
End Time: Apr 16, 2008 12:35:53 PM
```

If selected for output in the Model class, the files `population.txt` and `individuals.txt` written, the former containing selected statistical values for the whole population (for each run and each simulated year), the latter containing selected statistical values for each individual agent (for each run and each simulated year). Since both these files are produced by subclasses of the `AbstractStatistics` class, it is possible to utilise its generic selective output configuration mechanism, i.e. one may selectively annotated those variables with the `@Output` annotation which one wants to have calculated and written to the files.

3 References

North, M.J., Collier, N.T. and Vos, J.R. (2006). Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit, ACM Transactions on Modeling and Computer Simulation, Vol. 16, Issue 1, pp. 1-25, ACM, New York, New York, USA.