TUTORIAL: Programming agents in electronic institutions

Carles Sierra, Bruno Rosell, Juan A. Rodríguez-Aguilar, Josep Ll. Arcos

1. INTRODUCTION

This tutorial is based on an electricity market institution. Therefore, the student is expected to have background knowledge on the institution from the documentation on the EIDE distribution and the references below.

The purpose of this tutorial is manifold:

- To learn how to run electronic institutions over AMELI
- To learn how to simulate electronic institutions
- To provide a gentle introduction to agent programming in electronic institutions

2. INSTALLING/RUNNING THE ELECTRICITY MARKET

Before installing/running the electricity market institution the following software is required to be already installed in your system:

- *Java development kit* (version 6.0 at least) (http://java.sun.com)
- A*pache ant tool* (version 1.7.0 at least) *(http://ant.apache.org)*

Thereafter you only have to unzip the electricity market distribution file you can find at http://e-institutor.iiia.csic.es/eide/pub/ElectricityMarket-2.8.zip. Now you only have to open a terminal, change to the *etc* directory and execute the next command:

D:\YourHome\ElectricityMkt\etc> ant -p

And a list of the available targets will appear as a listing on your terminal:

- islander: to access the electricity market specification
- aBuilder: to run market sessions and simulations
- ejs: to start the Easy Java Simulator to specify an expected market demand
- ameli: to run the institution over the AMELI platform (run mode)
- oms: to run the institution over the OMS platform (simulation mode)
- jar: to compile the java classes
- javadoc: to generate the API documentation

Take the note above as informative. Throughout the exercises below we will be telling you the tools, out of the tools above, that you need to employ.

Integration with Eclipse

If you prefer to integrate the electricity market with Eclipse to run it and simulate it as well as to code the agents required in the exercise below, we recommend to stick to the following steps:

1. Open Eclipse
2. Create a new project (File -> New -> Project)
3. Select the ‘Java Project’ wizard
4. Name your project as ‘ElectricityMarket’ (or just make your own choice!). Next, in the content field select ‘create project from existing source’ and click on the *Browse* button to get to the directory where you have been unzipped the distribution file (ElectricityMarket-2.8.zip). After that click *Finish*.
5. Expand the *ElectricityMarket* project and the *etc* directory on the *Package explorer* left hand side pane. Click on the alternative button (righ button on your mouse) over ‘build.xml’. On the pop up menu select ‘RunAs -> Ant Build…’ (Don’t confuse it with the default ant option provide by Eclipse!). Now you must follow the next steps:
    a. Select the ‘classpath’ tab ant click on the ‘Ant home…’ button on the right hand side. Set the directory where ant 1.7.0 is installed in your system.
    b. Change to the ‘Targets’ tab and deselect any selected target
    c. Go to the ‘Main’ tab and set as argument the selected target (namely islander, aBuilder, ejs, ameli, oms).
    d. Click on the ‘run’ button.

Notice that, as mentioned above, throughout the exercises below we will be telling you the targets, out of the tools above, that you need to employ.

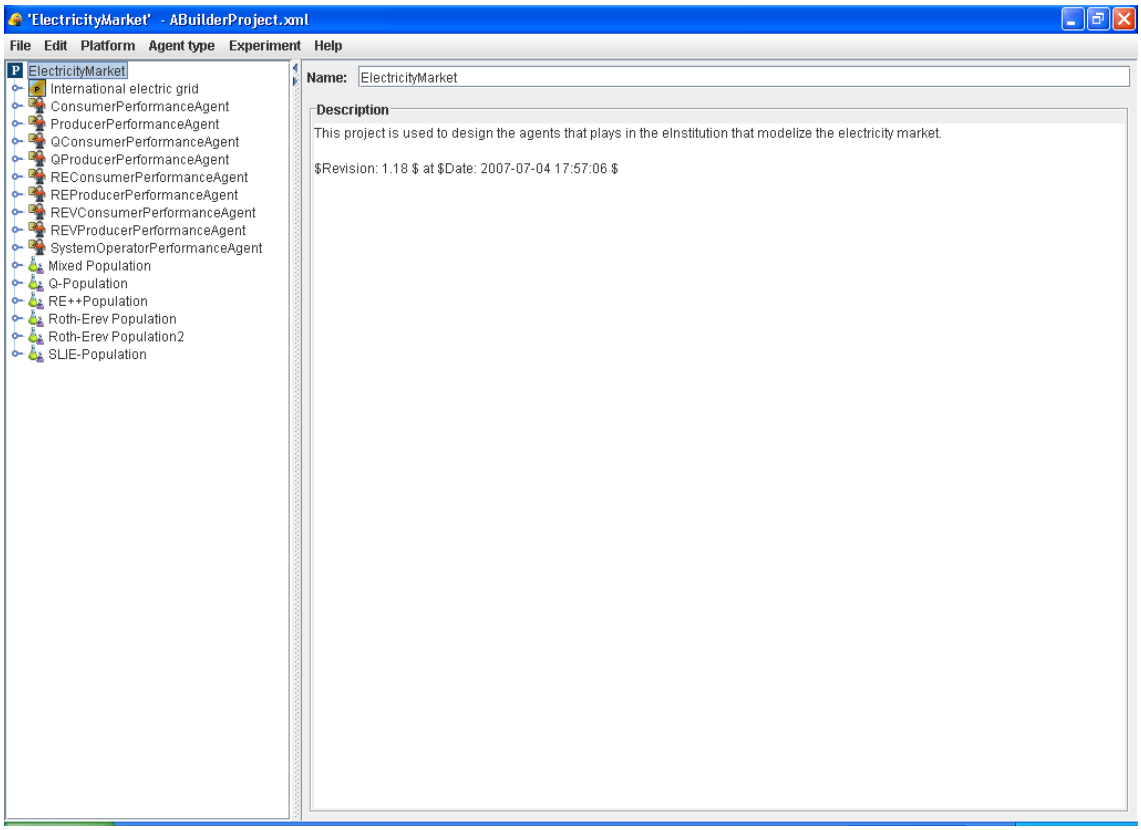
3. EXERCISES

Exercise 1

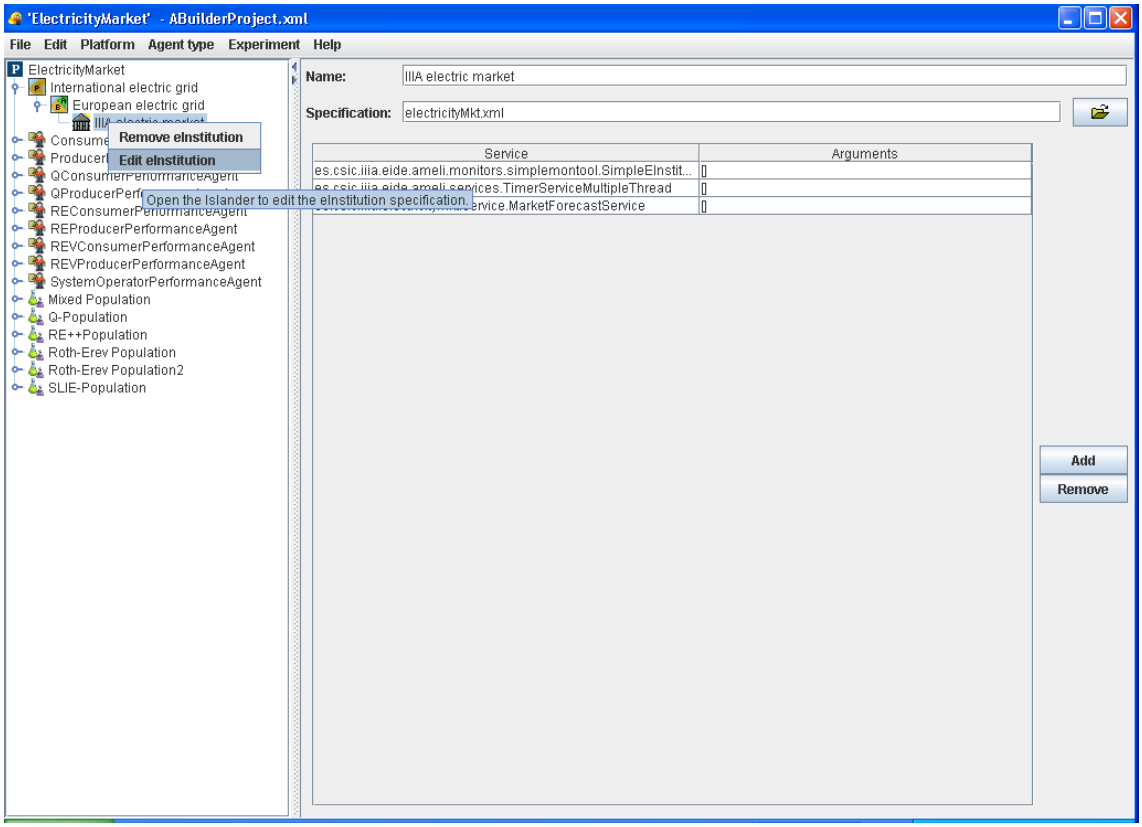
Start aBuilder. You only have to open a terminal, change to the *etc* directory and execute the next command:

D:\YourHome\ElectricityMkt\etc> ant aBuilder

The following GUI should appear on your screen:

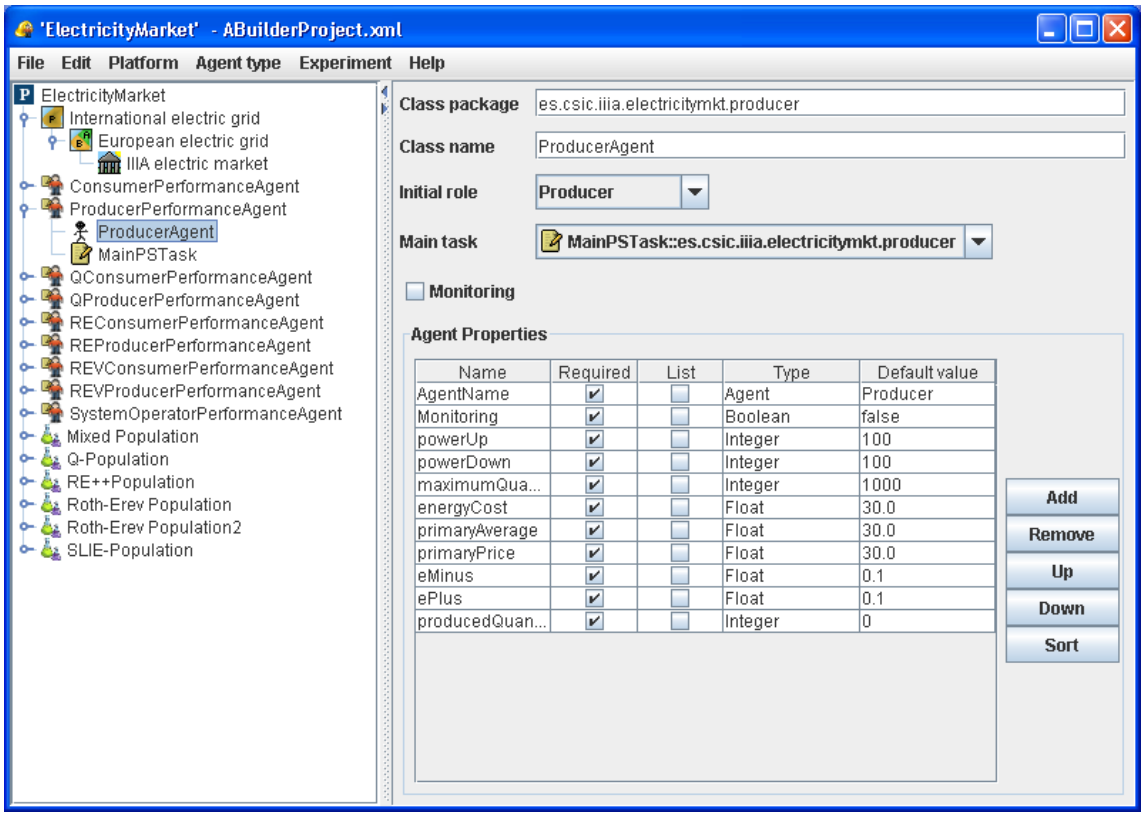


On the left hand side, expand the International electric grid icon, and subsequently the European electric grid icon, so that the IIIA electric market icon shows up. Click on this icon with the right button of your mouse and choose the Edit eInstitution menu option. This action shall start Islander with the electricity market specification. Take a few minutes to analyse the specification and make sure that you understand: (i) how the market operates; (ii) how producers and consumers interact with the rest of players in the market.



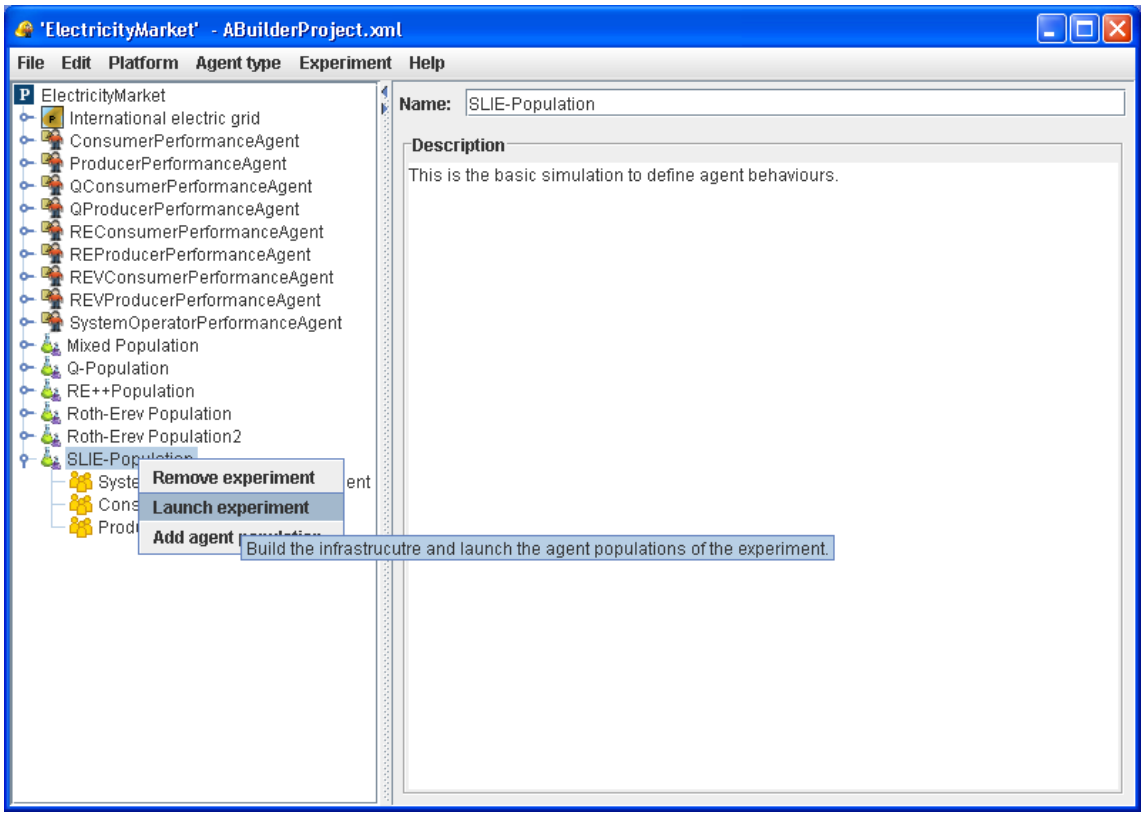
Exercise 2

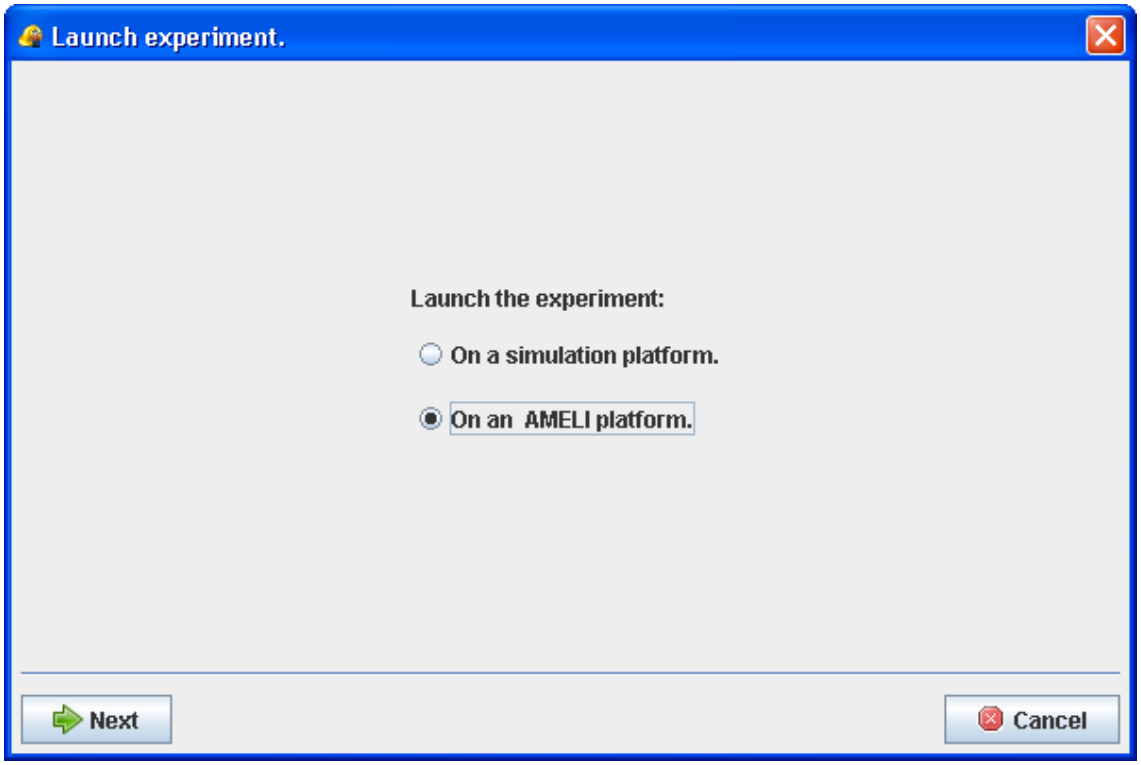
Back to aBuilder, on the left hand side pane expand the SLIE-Population icon. This icon stands for the specification of a simulation involving an agent population composed of several types of agents. By clicking on the SystemOperatorPerformanceAgent, ConsumerPerformanceAgent and ProducerPerformanceAgent (corresponding to agent types) you will see how agent populations are to be generated for the simulation. Notice that each one of the agent types chosen to define the simulation is defined above with the very same name. By expanding the icons corresponding to agent types you can access the properties of each agent type. The numerical properties are the ones that can be selected when specifying a simulation to generate varying agent populations.



Exercise 3

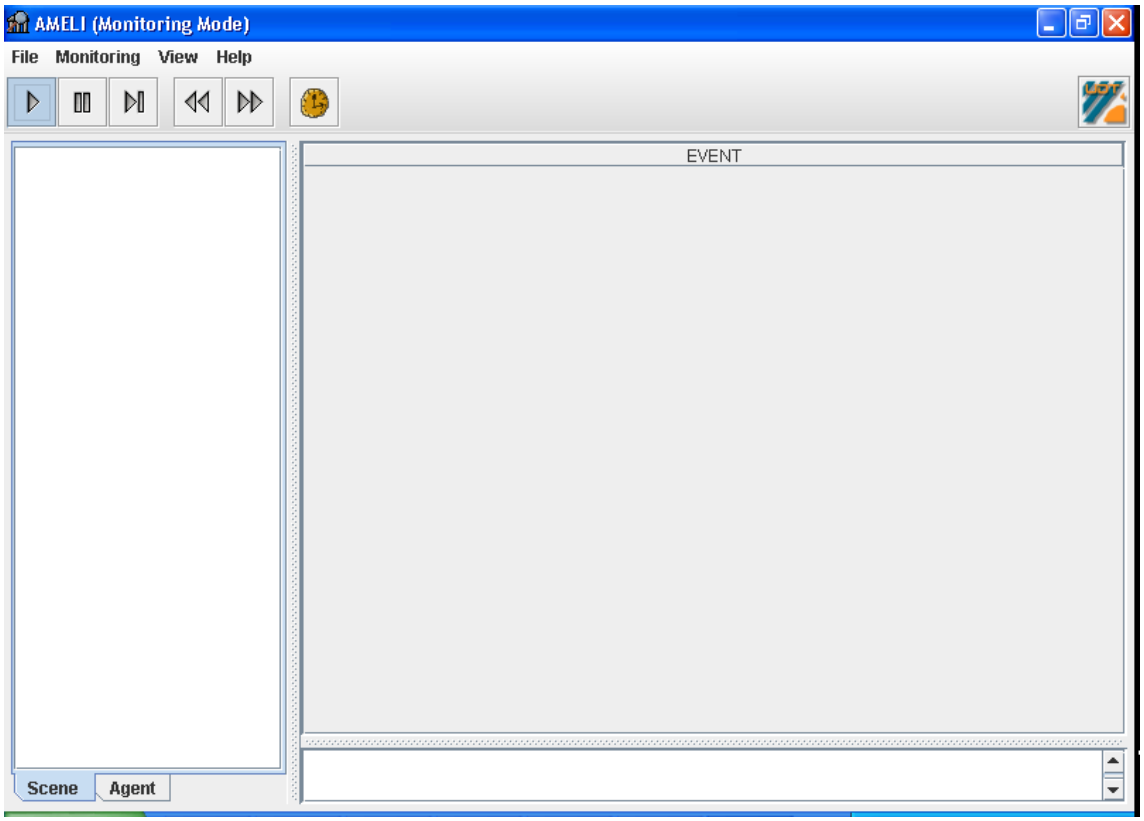
Click on the SLIE-Population icon with the right button of your mouse and select the Launch Population menu option. A wizard to start the simulation shall appear as shown below:



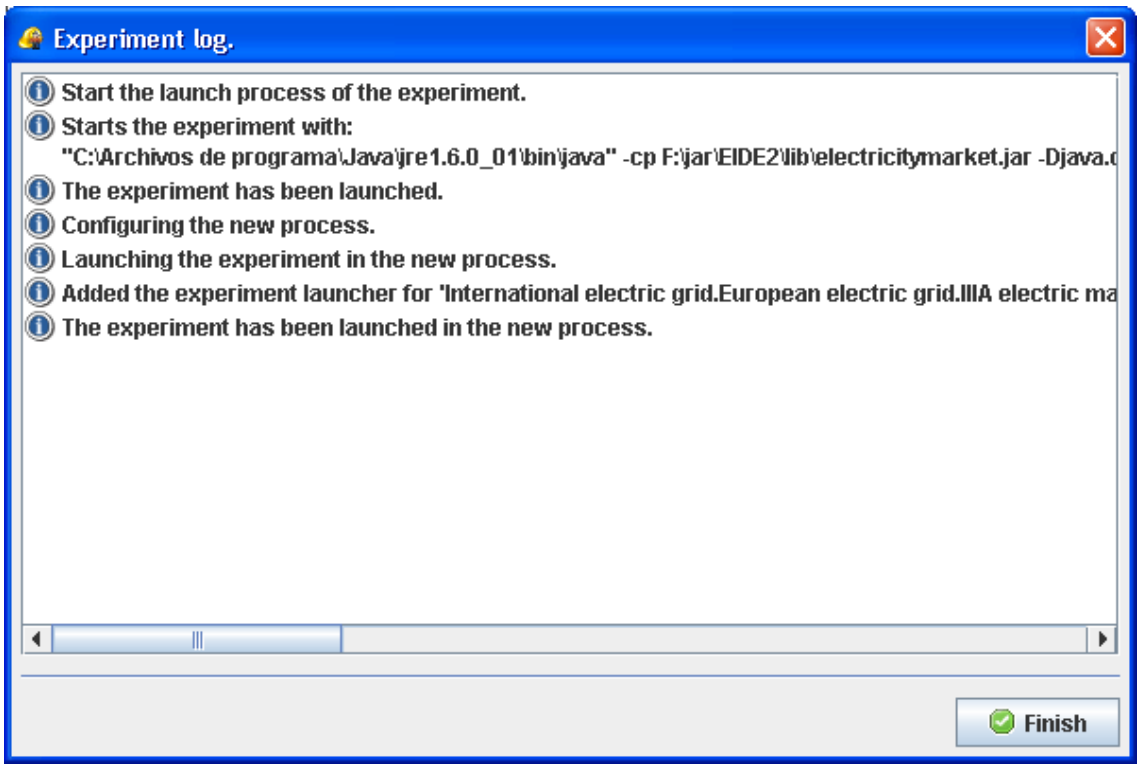


Click on Next twice to proceed with the run of the electronic institution. Several windows shall appear on your screen:

The AMELI monitoring tool displaying all the events occurring in the electricity market institution.

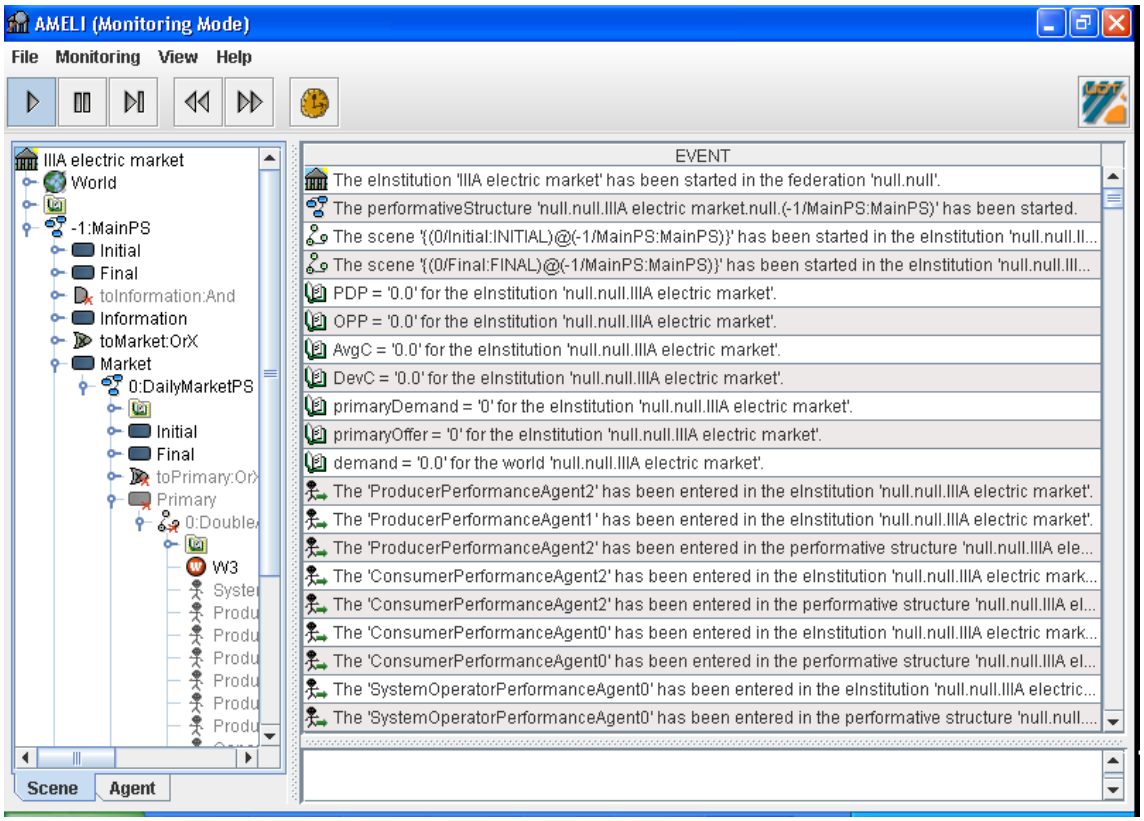


The aBuilder log.



The purpose of this exercise is to check that the participating agents in the electricity market behave according to the ISLANDER specification you have been analysing in exercise 2 (unlike exercise 2, you must not start ISLANDER from aBuilder, and instead

you must execute it in parallel with aBuilder by running **D:\YourHome\ElectricityMkt\etc> ant islander**). At this aim, select the play action in the OMS player window. Next, observe on the AMELI monitoring tool the dynamics of the market by expanding the icons on the left hand side pane. Notice that you can choose to either monitor the institution activity at the scene level or at the agent level. Notice too that the events occurring both at the scene and agent level are displayed on the right-hand side pane.

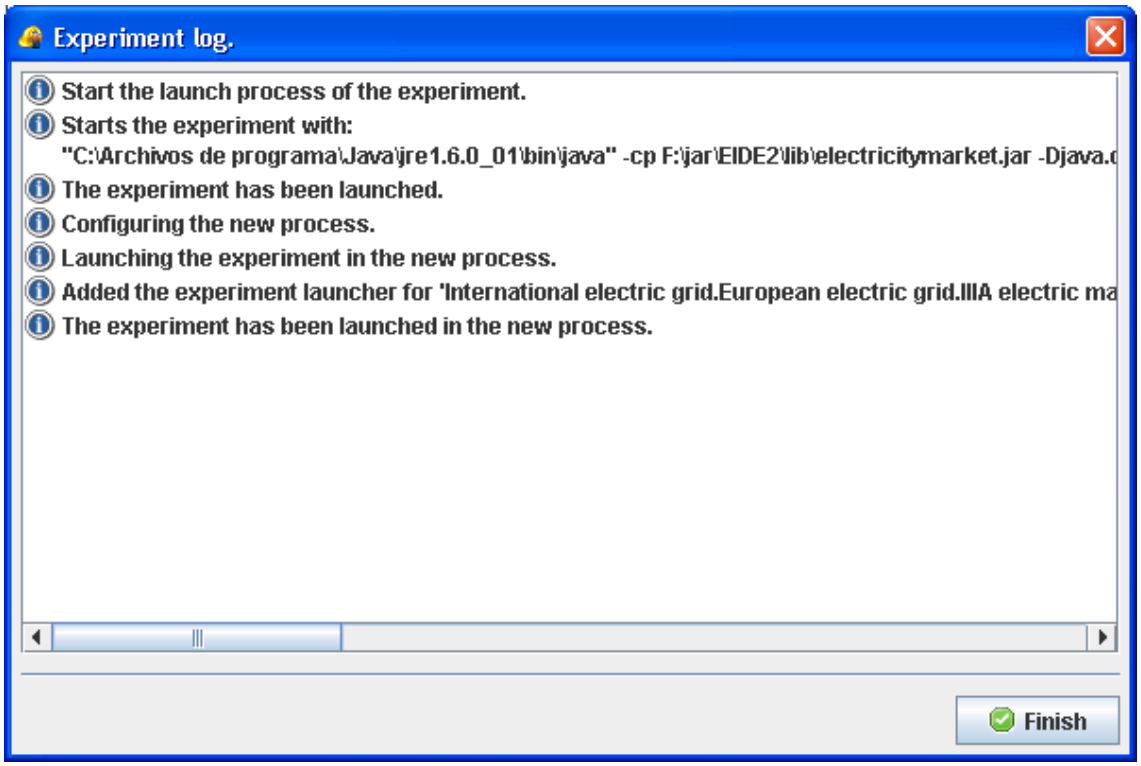


What are the properties of the environment (world) for the institution?
What are the institutional properties? In order to access their specification along with their meaning we suggest referring to the properties and description of the institution in the ISLANDER specification. Do the institutional properties change along the execution?

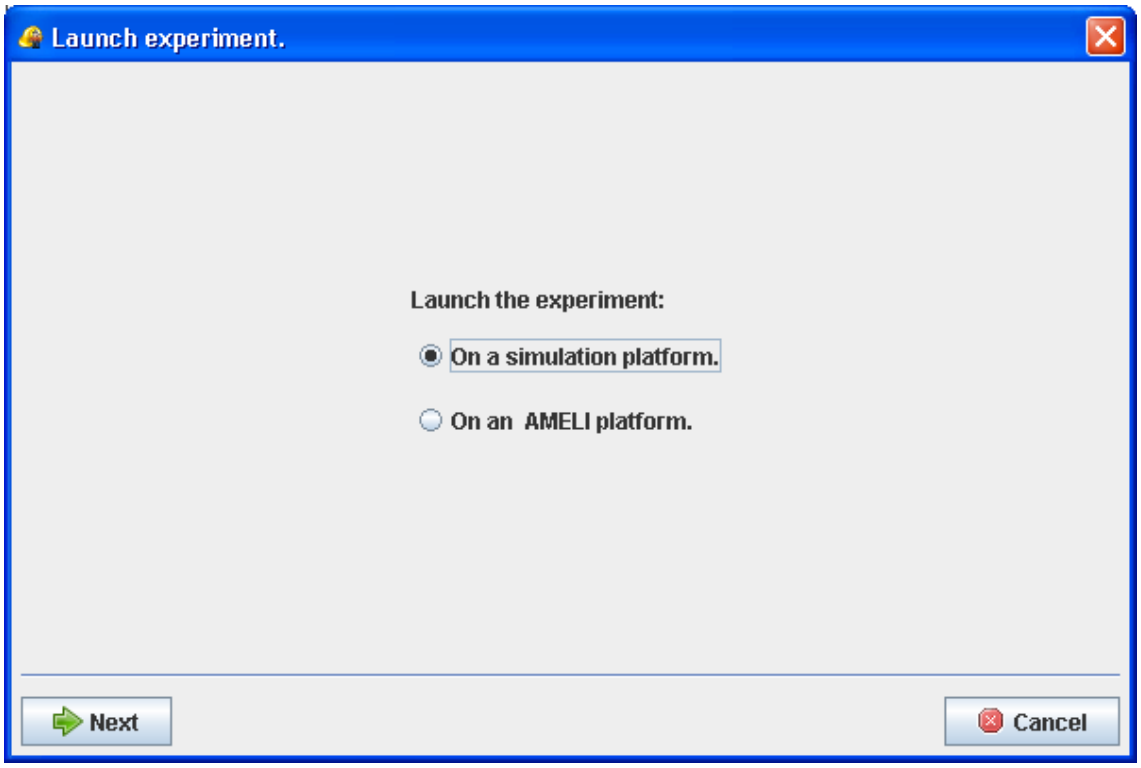
In order to finish the simulation simply click on the Finish button on the log window.

Exercise 4

Close aBuilder by clicking on the Finish button of aBuilder log.

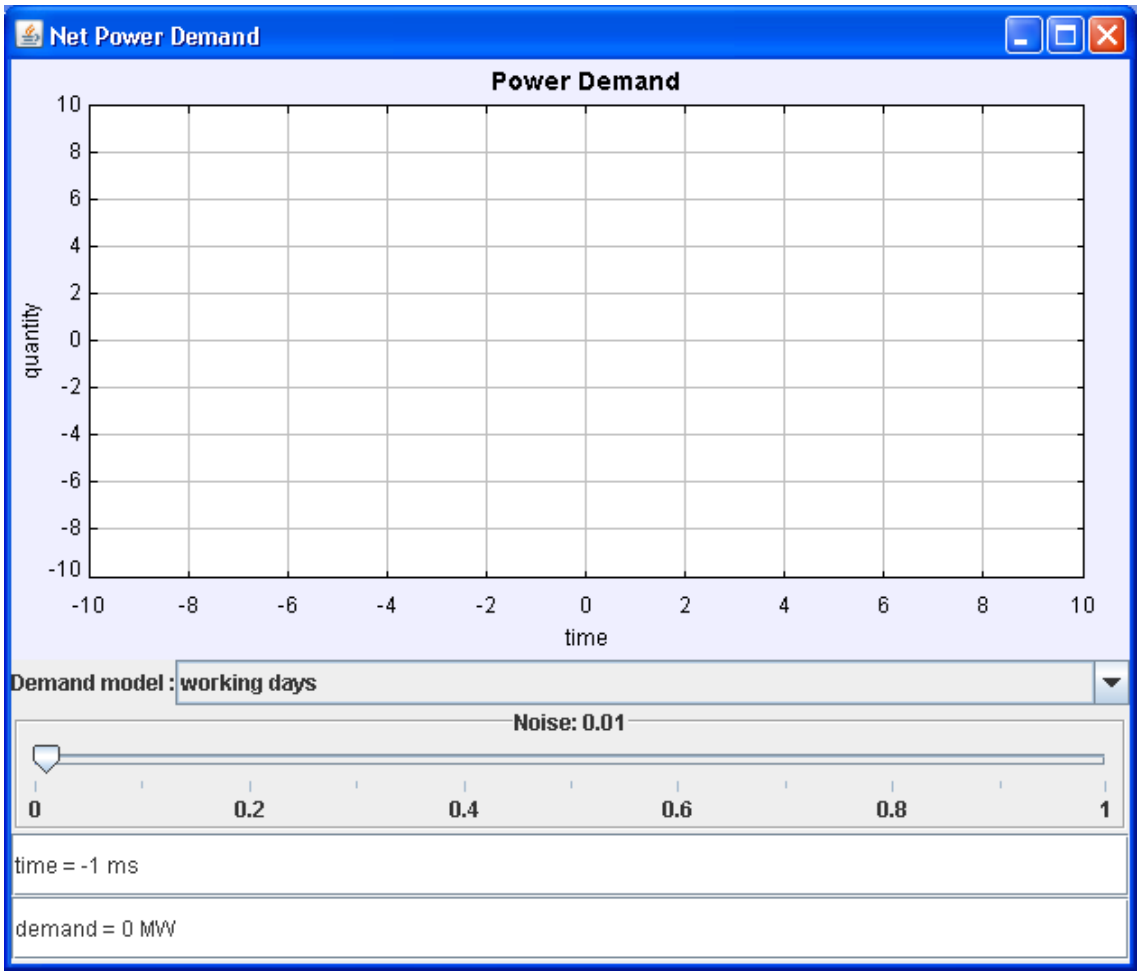


Now we are ready to start out the simulation of an electricity market. Start aBuilder as specified above. Select the SLIE-simulation button on the aBuilder GUI. Click on the right button of your mouse and select the Launch Population menu option. When the wizard shows up, choose to start the agent population on a simulation platform (as a simulation).

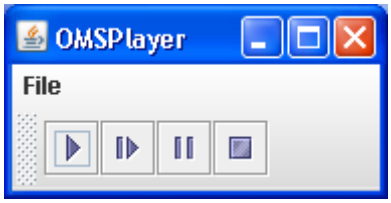


When accepting the simulation configuration, aside from the AMELI monitoring tool you played with in the preceding exercise, there is a new collection of GUIs.

The Net Power Demand window displaying the simulation of the actual-world market demand.

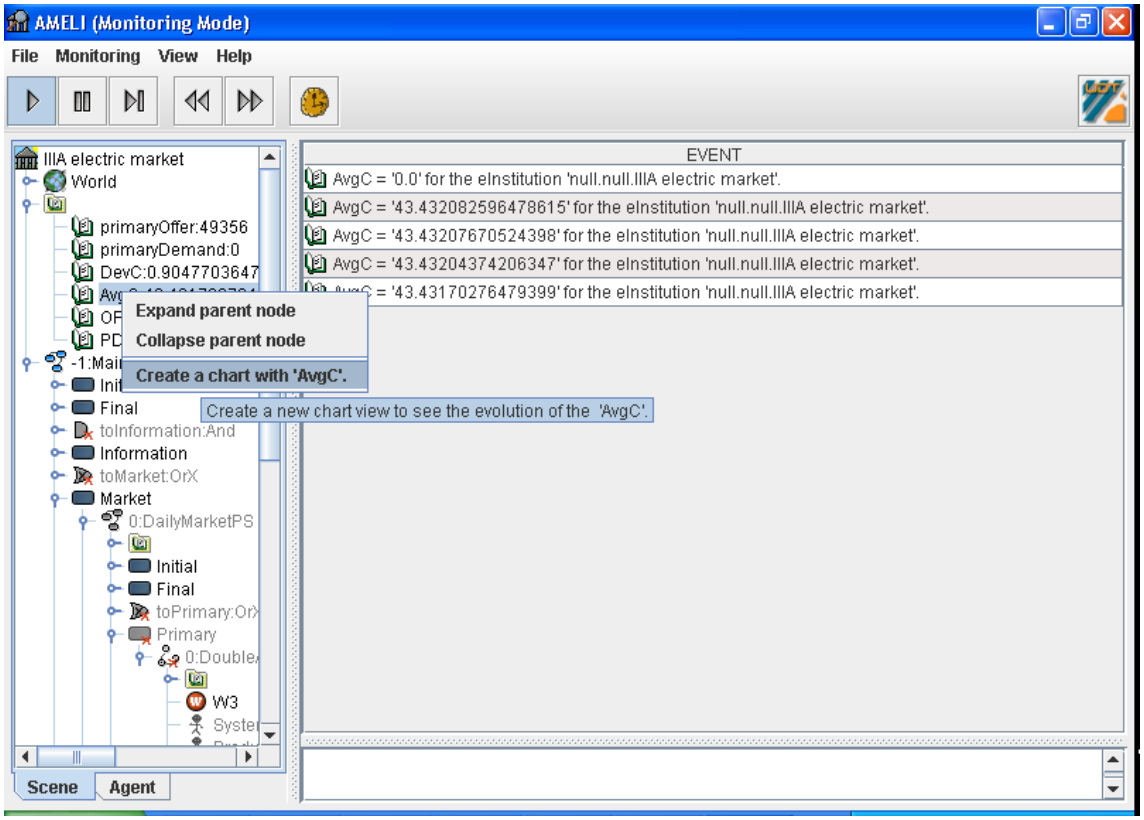


The OMS player window containing the simulator controls, namely play, step, pause, and stop respectively.

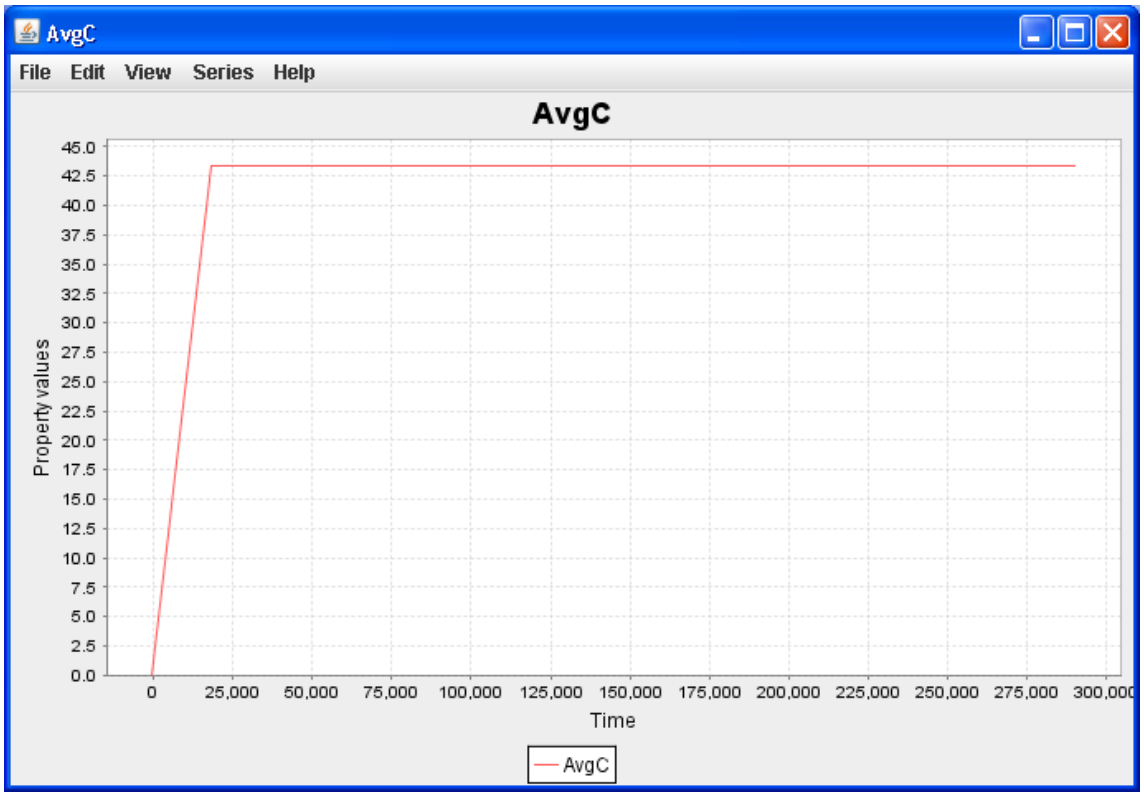


Start the simulation by clicking on the *Play* icon of the OMS interface above.

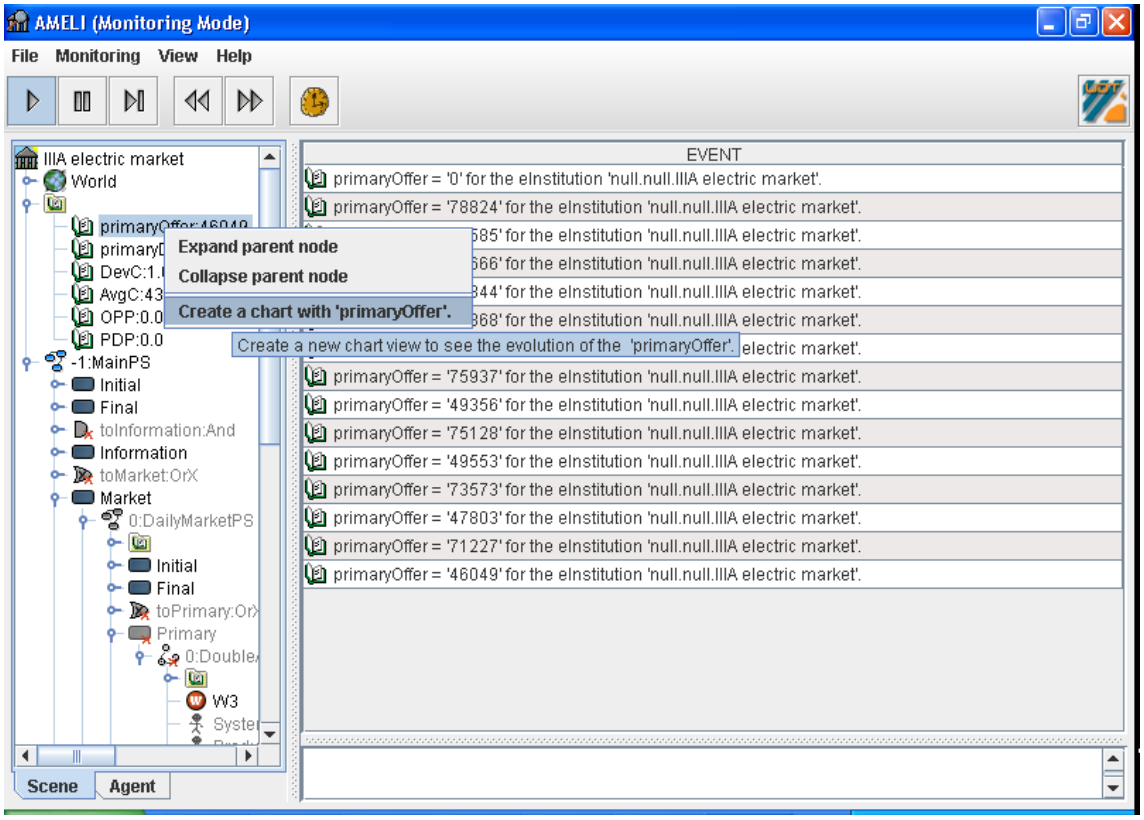
Generate a plot for the institutional property power average cost (AvgC) as follows:

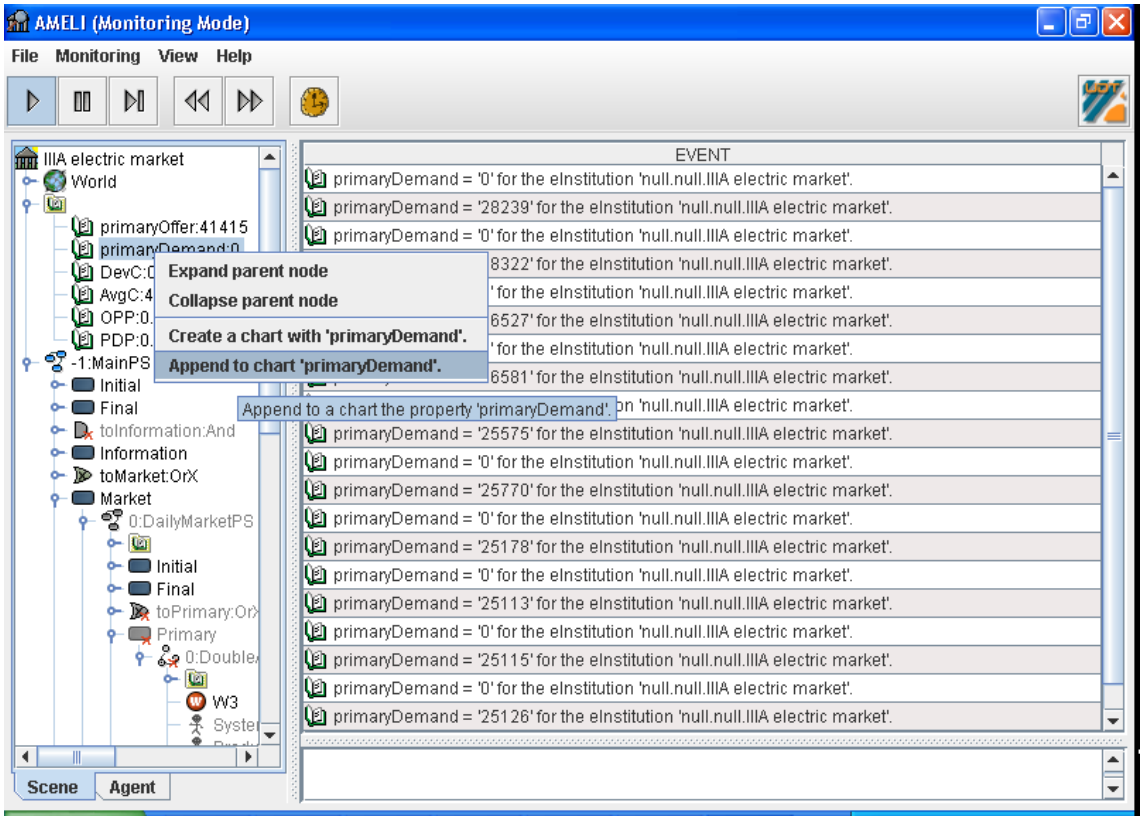


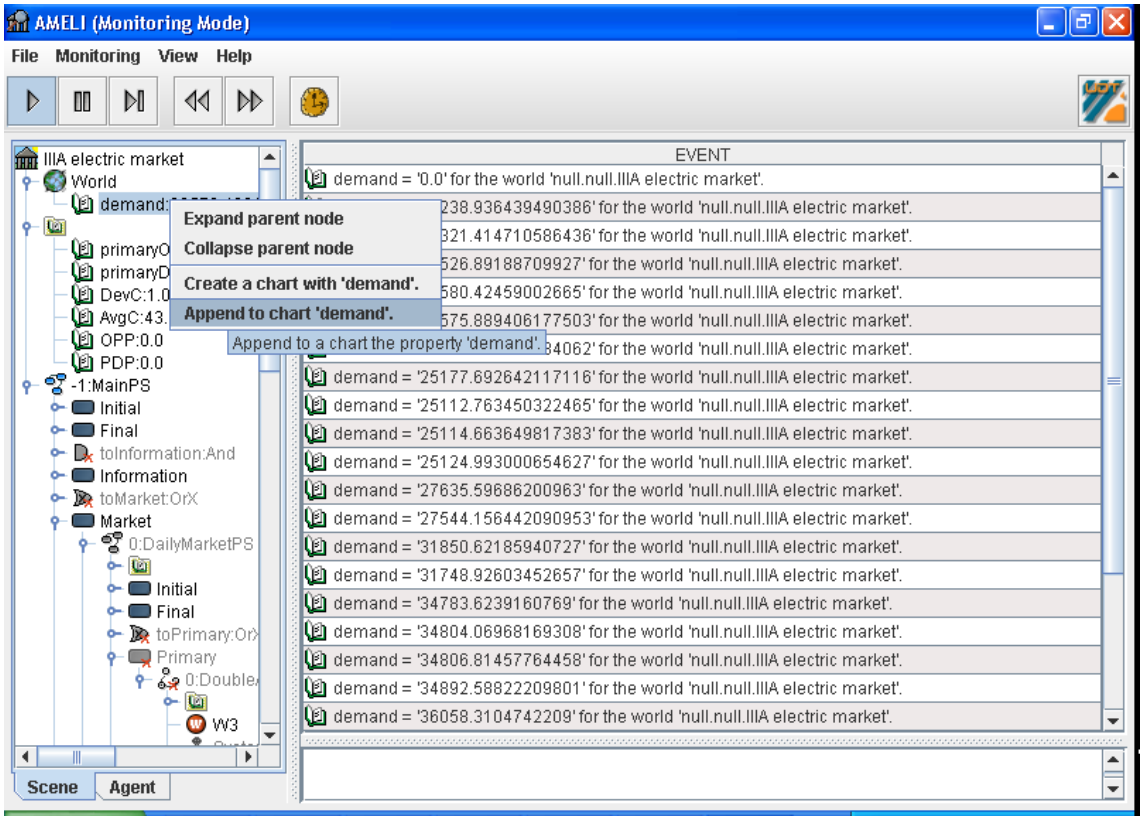
As a result the following display should come up:



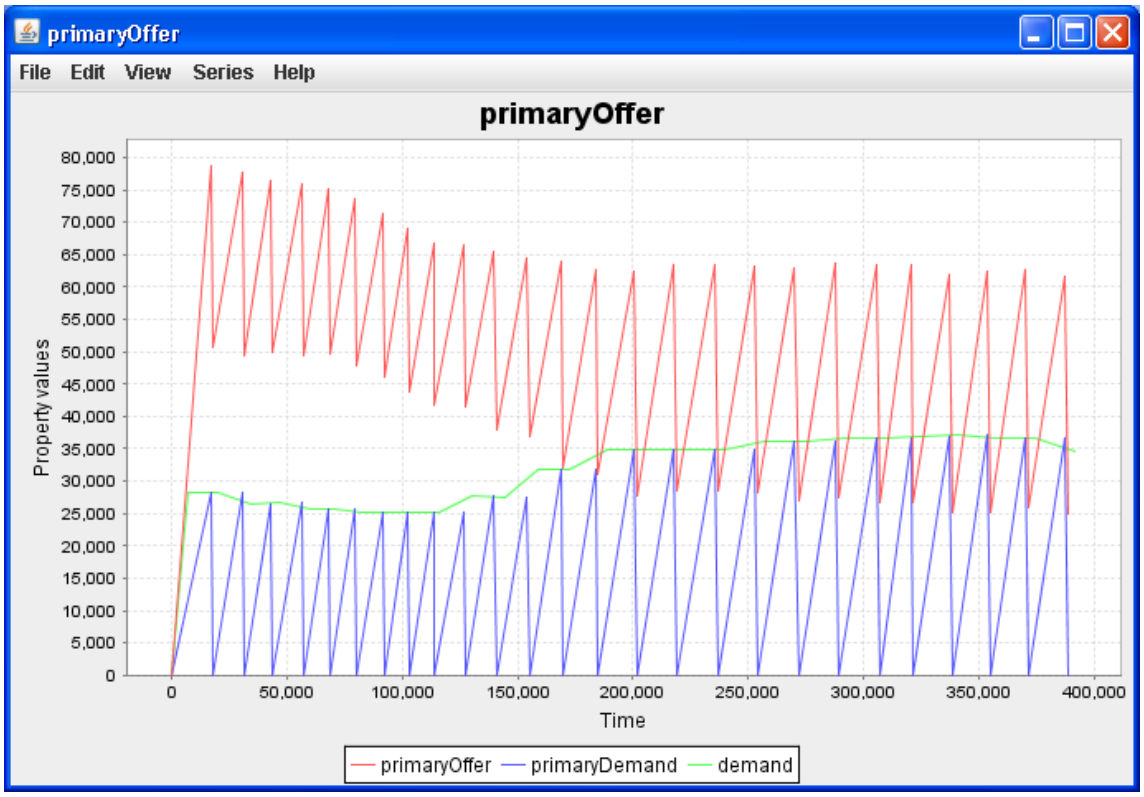
Generate another plot for the offer (primaryOffer) and demand (primaryDemand) institutional properties in the double auction market, as well as the forecast demand generated by the EJS simulator, as follows:







The result should be a plot for both demand and supply in the double auction market.



Notice that both plots show lots of up-and-downs. The highest peaks correspond to the overall offer and demand before running the double auction protocol, whereas the lowest peaks correspond to the overall remaining offer and demand after the double

auction protocol. Therefore, the differences between the highest and lowest peaks tell us the amount of energy sold and bought during a double auction.

Now take your time to answer the following questions:

- Do you observe whether the average power cost (AvgC) stabilizes or else suffers from oscillations?
- What is the percentage of energy sold in the double auction market?
- What is the effect of changing the demand model from working days to another model (namely holidays, Saturdays, and Sundays)?
- If the power cost average stabilizes, do you observe whether the strategies of producers and consumers in the double auction significantly change?

Exercise 5

The SLIE-Population on the aBuilder GUI defines a market with more producers (offer) than consumers (demand). Define now a *scarce market* with less producers (offer) than consumers (demand) (we recommend 3 producers and 10 consumers with the same features as the SLIE-Population). For this purpose you can either modify SLIE-Population of create a new population by choosing the Add experiment option out of the Experiment menu option.

- Is there any blackout in the market? If so, how would you avoid them?
- Does the average power cost (AvgC) change when there is more demand than offer?
- What is the percentage of energy sold in the double auction market?
- What is the effect of changing the demand model from working days to another model (namely holidays, Saturdays, and Sundays)?

Exercise 6

Define now a balanced market involving as many producers as consumers (we recommend 5 producers and consumers with the same features as the SLIE-Population).

- If the system operator required that the safety power of producers is less than 10%, do you reckon that there would be any blackouts? Hint: Plot the value of the OPP institutional property, namely the percentage of energy that the system operator can enforce producers to generate in order to fully satisfy the market demand.
- Out of the three types of markets ran so far, and taking into account that the features of producers and consumers in the three of them are the same, which one is the less risky?

Exercise 7

It is time to code your own agent. The purpose of this exercise is to code a myopic producer that offers all the energy he can produce at the price a SLIE producer would offer energy. For this purpose, we suggest to modify the MyQDoubleAuctionPerformance class that implements the behaviour of a producer agent implemented by the MyQProducerAgent class. Both classes are located in the src/java/es/csic/iiia/electricitymkt/producer directory. You can refer to the doc/api

directory to access the API of the MyQProducerAgent. We suggest implementing the following function to assess the quantity to offer:

$$q_i(P) = \left\{ \begin{array}{cc} CS_i, & \text{if } P > \alpha_i \\ [0, CS_i], & \text{if } P = \alpha_i \\ 0, & \text{if } P < \alpha_i \end{array} \right\}$$

where P stands for the selected price, α_i stands for the producer's cost, and CS_i stands for the producer's maximum capacity. Notice that by implementing the function above the producer would offer as much energy as possible as long as the selected price is above his cost, otherwise he would offer a random value when the selected prices equals the cost and 0 if losses may come about.
Analogously, you can implement the double auction behaviour for a consumer by changing the MyQDoubleAuctionProtocol class of the MyQConsumerAgent class in the src/java/es/csic/iiia/electricitymkt/consumer directory.

How to compile your agent to include it in aBuilder? There are two shorcuts:

- *Out of Eclipse:* Close aBuilder and reopen it (running "ant aBuilder"). This will cause the compilation of the new source.
- Within Eclipse: Select 'Ant Build' in the 'Run as' menu option.

Exercise 8

Taking inspiration on the SLIE-Population, define a new experiment where the SLIE producers are replaced by the producers you have just implemented in the previous exercise. Experiment with the three types of markets described above and observe the effect of the new double auction strategy in the market. Do you think that this type of producers adapt to the market conditions?

Exercise 9

In a series of studies, Roth and Erev [1] have sought to understand how people learn individually to behave in games with multiple strategically interacting players. To this end, they have developed a three-parameter reinforcement learning algorithm, referred to as the RE algorithm. In [2] the authors propose to extend the work in [1] to implement buyers and sellers in a discriminatory double-auction electricity market. The purpose of this exercise is to code a myopic producer that prices his energy according to the strategy described in [2]. For this purpose, we suggest completing the MyREProducerAgent class located in the src/java/es/csic/iiia/electricitymkt/producer directory. You can refer to the doc/api directory to access the API of the MyREProducerAgent. We suggest implemeting the following functions to assess the probability of the price to offer:

$$q_{jk}(n+1) = (1-r)q_{jk}(n) + E(j,k,k',n,K,e)$$

where

$$E(j,k,k',n,K,e) = \begin{cases} R(j,k',n)(1-e), & k = k' \\ q_{jk}(n)\frac{e}{K-1}, & k \neq k' \end{cases}$$

The definitions of both functions are described in [2] and it is part of the introduction to this tutorial.
Analogously, you can implement the double auction behaviour for a consumer by completing the MyREConsumerAgent class in the src/java/es/csic/iiia/electricitymkt/consumer directory.
In [2] the authors suggest the following values for the reinforcement learning algorithm: (i) s(1) = 1.00, r=0.04, e=0.97 for 1000 auction rounds; (ii) s(1) = 1.00, r=0.02, e=0.99 for 10000 auction rounds.

Exercise 10

It is time to test the agents you have just implemented. Define a new experiment where the SLIE producers are replaced by the producers you have just implemented in the previous exercise. Experiment with the three types of markets described above and try to provide an answer to the following questions:

- Do you observe whether the average power cost (AvgC) stabilizes or else suffers from oscillations?
- After observing the behaviour of producing agents in the double auction, what can you say about their degree of aggressiveness? Do they try to continuously maximize their profits or else they adopt a conservative behaviour?

Exercise 11

You may have noticed that producers and consumers inspired on [2] are not very aggressive. It takes a long time for then to adapt after they lose in a double auction. We propose to slightly modify the agents you coded in exercise 10 according to the following rules:

- If an agent sells all the offered quantity, instead of reinforcing the winning price, he does reinforce a higher price because he considers the sale as an indicator of an increase in market demand.
- If an agent does not manage to sell all the offered quantity, he reinforces lower prices because he considers that either: (i) the market demand is going down; or (ii) his competitors are going cheaper.
- If an agent does not succeed in selling energy, he shifts the probabilities for the prices he has been using for winning prices to reinforce cheaper prices, and thus increase his probability of succeeding in the next round.

For this purpose, we suggest completing the code of the MyREVProducerAgent class. This class is located in the src/java/es/csic/iiia/electricitymkt/producer directory. You can refer to the doc/api directory to access the API of the MyREVProducerAgent.

Exercise 12

It is time to test the agents you have just implemented. Define a new experiment where the SLIE producers are replaced by the producers you have just implemented in the previous exercise. Experiment with the three types of markets described above and try to provide an answer to the following questions:

- Do you observe whether the averge power cost (AvgC) stabilizes or else suffers from oscillations?
- What is the effect of changing the demand model from working days to another model (namely holidays, Saturdays, and Sundays)?
- What happens in terms of AvgC in the three market configurations (scarce, balanced, overproduction)?
- What do you observe concerning AvgC if you change the risk factor of producing agents (e.g. try with 20% and 30%)?
- Observing the behaviour of producing agents in the double auction, what can you say about their degree of aggressiveness? Do they try to continuously maximize their profits or else they adopt a conservative behaviour?

Exercise 13

Can you design and implement bidding strategies for producing agents that outperform (make more profit than) the producing agents you encoded in exercise 11?

Exercise 14

Do you think that aBuilder is a useful tool? What for? What kind of applications can you think of this tool may provide support for?

References

[1] *Ido Erev* and *Alvin Roth*. Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria. *American Economic Review*, 1998, vol. 88, issue 4, pages 848-81.
[2] J. Nicolaisen, V. Petrov, and L. Tesfatsion. *Market power and eciency in a computational electricity market with discriminatory double-auction pricing*. IEEE Transactions on Evolutionary Computation, 5(5):504-523, October 2001.