DSL3S – Domain Specific Language for Spatial Simulation Scenarios

Simple Test Session – User Guide, v1.0, 2015

The main goal of this User Test Session is to perform an evaluation of DSL3S (language and framework) by users not familiarized with it, in order to assess usability and detect eventual bugs or other limitations. This evaluation will be performed through the development of a very simple case study with a Predator-Prey model. The results gathered will be used to assess the language usability as well as its future enhancements. The case study application is described below.

Case Study – Predator-Prey simulation

This simulation evolves in a space covered with pasture feedstock of variable volume, available from a raster dataset. The feedstock replenishes itself slowly at each time step.

A prey animat wonders around grazing the available pasture. Prey spends one energy unit moving at each time step and stores the energy gazed from the pasture. When it reaches a certain high amount of stored energy, the prey can reproduce creating a new prey that inherits part of the energy stored by its parent. If the energy stored by a prey falls down to 0 it is discarded from the simulation. Initial locations of prey, as well as stored energy, are available from a vector dataset.

At simulation start a number of predator animats are cast randomly over the pasture space. They possess an energy store that they spend as they move. They seek prey, taking all their stored energy when captured (i.e. when in the same location). Like prey, predators replicate themselves when their energy store raises above a certain threshold and perish if it falls to 0.

An additional vector layer portraits areas where prey cannot go, limiting their movements. In these areas the pasture grows freely and is left untouched.

Conditions:

This User Test Session will be conducted under the following conditions:
- Realization of the task without previous knowledge on DSL3S;
- The user must have a computer running Eclipse Luna and Java (version 6 or higher);
- While users perform the assigned task, their behavior and performance will be registered;
- Performance criteria like the time spent and number of errors made will be taken into account;
- Users can think out loud and share ideas if they want;
- The evaluator does not interact with the users until the tests are finished (except in case of blocking errors);
- The test itself is planned to last 30 minutes (at most).

Setup

1. Follow the instructions in the DSL3S Wiki to install the required plug-ins:
   https://github.com/MDDLingo/DSL3S/wiki/Installation
2. Download the Java libraries (MASON, JTS, etc) as indicated in the Wiki.
3. Download the sample datasets from the following URL:
   http://mddlingo.github.io/DSL3S/data/demoData.zip

Instructions:

4. Create a new Java project in Eclipse with the project wizard.

   4.1 Add a source folder called *src-gen* to the project (*Source* tab);

   4.2 Add references to the Java libraries downloaded before (*Libraries* tab > *Add External JARs...*);

   4.3 Keep (or switch to) the Papyrus perspective.

5. Create a new folder in the project called *data.* Expand the datasets package in the file system and copy its contents into the new *data* folder.

6. Create a new Papyrus model (*File* menu > *New* > *Papyrus Model*)

   6.1 Name it "SimpleTest.di" and make sure the "SimpleTest" project is selected; click *Next>*.

   6.2 Select *UML* as language and *Class Diagram* for the first diagram; name it "Simulation".

7. Click on the diagram space and in the *Properties* view select the *Profile* tab. Click the registered profile button (⬆) and then select DSL3S in the following menus.

8. Add a new class to the diagram and give it a suggestive name such as "TestSim" or "MySimulation".

   8.1 With the new class element selected, access the *Profile* tab in the *Properties* view and click the stereotype application button (➕). Select the *Simulation* stereotype in the following menu, add it to the *Applied Stereotypes* box with the arrow buttons and click *OK*.

   8.2 In the *Profile* tab expand the *Simulation* item to view its properties.

   8.3 Click on the *simulName* property and in the text input box to the left type something like "Predator-Prey prototype".

   8.4. Edit *spaceWidth* and *spaceHeight*, setting both to "100".

9. Add a new class to the diagram, naming it "Pasture".

   9.1 Apply the *Spatial* stereotype on it; set the *inputLayer* property to "data/Pasture.agrid" and *stepVariation* to "0.5".

   9.2 Set *colourMin* to "224,224,128" and *colourMax* to "32,128,32".

   9.3 Link "Pasture" to the *Simulation* element with an association edge.

10. Add a new class to the diagram, and name it "Prey".

    10.1 Apply the *Animat* stereotype on it and set the *inputLayer* property to "data/Prey.shp" and *wanderer* to "true".

    10.2 Set *colourMin* to "64,64,255" and *colourMax* to "32,32,186".

    10.3 Link "Prey" to the *Simulation* element.

11. Open the *Model Explorer* view and right click on the root of the model tree it shows. Select *New > Create new UML Class Diagram* and name it "Prey".

    11.1 Drag-and-drop the "Prey" element from the model tree into this new diagram.

    11.2 Drag-and-drop the "Pasture" element in the same way.

12. Create a new class, name it "PreyEnergy".

    12.1 Apply the *Attribute* stereotype on it, set the *layerAttribute* property to "Energy", the *stepVariation* attribute to "-1" and *maxValue* to "50".

    12.2 Associate "PreyEnergy" to the "Prey" element.

13. Create a new class and name it "Graze".

    13.1 Apply on it the *Harvest* stereotype, and set the *percentHarvested* property to "100".

    13.2 Associate "Graze" with "PreyEnergy" and then with "Pasture".

14. Save the model and generate code from it.

    14.1 In the *Package Explorer* view expand the "SimpleTest" item and right-click the *uml* element, select *MDD3S > Generate simulation from DSL3S model*.

    14.2 Expand the *src-gen* folder and search for the GUI class (e.g. *SimpleSimGUI.java*). Right click and select *Run As > Java Application*.

    14.3 In the simulation window click the play button and observe the prey (blue dots) wandering and grazing the pasture.

15. Add a new class to the "Prey" diagram and call it "PreyReplicate".

    15.1 1.1. Apply the *Replicate* stereotype and set *lowerTreshold* to "-1", *upperTreshold* to "30", toll to "10" and *inheritance* also to "10".

    15.2 Associate "PreyReplicate" to the "PreyEnergy" attribute.

16. Add another, class calling it "PreyPerish".

    16.1 Apply the *Perish* stereotype and set *lowerTreshold* to "0" and *upperTreshold* to "51".

    16.2 Associate "PreyPerish" to the "PreyEnergy" attribute.

17. Generate again the code and run it. Observe the prey replicating and rapidly grazing the pasture.

18. Switch back to the "Simulation" diagram and add a new class, named "Predator".

    18.1 Apply the *Animat* stereotype, set *wanderer* to "true" and *initNum* to "25".

    18.2 Set *colourMin* to "255,64,64" and *colourMax* to "192,0,0".

    18.3 Associate "Predator" with the *Simulation* element.

19. Create a new diagram named "Predator" and drag-and-drop "Predator" and "PreyEnergy" into it.

20. In the new diagram add a new class called "PredEnergy".

    20.1 Apply the *Attribute* stereotype setting *initValue* to "60", *maxValue* to "100" and *stepVariation* to "-1".

    20.2 Associate "PredEnergy" with "Predator".

21. Add another class called "PredReplicate".

    21.1 Apply the *Replicate* stereotype and set *lowerTreshold* to "-1", *upperTreshold* to "70", toll to "30" and *inheritance* also to "30".

    21.2 Associate "PredReplicate" to the "PredEnergy" attribute.

22. Add yet another class called "PredPerish".

    22.1 Apply the *Perish* stereotype and and set *lowerTreshold* to "0" and *upperTreshold* to "101".

    22.2 Associate "PredPerish" to the "PredEnergy" attribute.

23. Instruct "Predator" to seek for prey, add another class and call it "Seek".

    23.1 Apply the *Move* stereotype, set *weight* to "1" and *scope* to "1.5".

    23.2 Associate "Seek" with "Predator" and then with "PreyEnergy".

24. Create a new diagram named "AnimatInteractions"; drag-and-drop the "PredEnergy" and "PreyEnergy" elements.

25. In the new diagram add a new class named "FeedPredator".

    25.1 Apply the *Supply* stereotype and leave all properties by default.

    25.2 Associate "FeedPredator" to "PreyEnergy".

26. Add another class named "EatPrey".

    26.1 Apply the *Harvest* stereotype, set the *percentHarvested* property to "100" and *scope* to "0.5".

    26.2 Associate "EatPrey" to "PreyEnergy".

27. Associate "FeedPredator" with "EatPrey"; "Predator" can now feed itself.

28. Generate the code again and run it. Observe predator animats feeding off the excess of prey.

29. Switch to the "Simulation" diagram and add a class named "Inacessible".

    29.1 Apply the *Spatial* stereotype and set inputLayer to "data/Protected.shp".

    29.2 Set *colourMin* and *colourMax* both to "160,160,160".

    29.3 Associate "Inaccessible" with "Simulation".

30. Switch now to the "Prey" diagram and drag-and-drop the "Inaccessible" element.

31. Create a new class named "Avoid".

    31.1 Apply the *Move* stereotype, setting *weight* to "-1000" and *scope* to "1.5".

    31.2 Associate "Avoid" to "Prey" and then to "Inacessible".

32. Create a new class named "Prefer".

    32.1 Apply the *Move* stereotype, setting *weight* to "1" and *scope* to "1.5".

    32.2 Associate "Prefer" to "Prey" and then to "Pasture".

33. Generate the code once again and observe that prey animats now avoid the polygons representing inaccessible areas.

34. If the "Inacessible" areas are not visible they are probably hidden by the "Pasture" *Spatial* element.

    34.1 Transparency can be set on "Pasture", adding a fourth parameter to the *colourMin* and *colourMax* properties, e.g. to "224,224,128,128".

Please, fill the "DSL3S Questionnaire" available online:

http://goo.gl/forms/LQMClXU834

Further Info:
- Wiki: https://github.com/MDDLingo/DSL3S/wiki
- The article: de Sousa, L. and da Silva, A. R., "Preliminary Design and Implementation of DSL3S", CAMUSS - International Symposium on Cellular Automata Modelling for Urban and Spatial Systems, Oporto, 2012.
  http://isg.inesc-id.pt/alb/static/papers/2012/C115-ls-CAMUSS-2012.pdf

Thank you for participating!