Delivery Fleet Optimization with GIS Functionality Webinar

May 28, 2015

Tom Baggio

1. Problem Definition
The webinar objectives is a step-by-step process of building a simulation model of an aircraft part manufacturer servicing a network of regional airports. We also extend the simulation to include an optimization experiment to determine the ideal number of vehicles required to maintain a targeted fleet utilization.

The model is comprised of several agents consisting of a network of regional airports, a single manufacturer, a fleet of delivery trucks, and orders. The model utilizes the integrated GIS functionality to locate the agents in the model as well as automatically determine the routes based on the roads from the GIS provider. Orders are created by the airports and received by the manufacturer. These orders are represented as agents and communicate between the Airport and the Manufacturer. After receipt of an order, loading the truck is simulated with a process logic embedded within the manufacturing agent. Once the truck is loaded, the order is delivered to the requesting airport and unloaded and sent back to the Manufacturer.

2. Model Framework
For this simulation we will perform the following steps:

1. Set locations of the airports and manufacturer using the GIS capabilities of AnyLogic. The routes will automatically be generated from the same GIS functionality.
2. Establish the process of ordering new parts and communicate/transfer the orders between the airports and manufacturer agents.
3. Build the manufacturer’s logic including order processing, truck loading and unloading, delivery receipt notification and truck return.
4. Create an optimization experiment to establish the optimal number of trucks with a utilization target of the maximum value of 85%.

3. Create New Model

1. Select file/New/model on the toolbar and enter “AirportDelivery”.
2. The dialog box will display the New Model properties, java package name, location, model name, and units. You can change the time units later.

![New Model dialog box](image)

*Figure 1: New Model dialog box*

Click Finish and the window opens to the agent Main where you will begin to create the model.
4. **Map Region Selection and Agent Locations**

Begin by using the **GIS Map** object from the **Space Markup pallet**, drag it onto the **Canvas** and adjust its size relative the model viewing area as seen in **Figure 2**.

![Image of GIS Map Object](image)

**Figure 2: Display of the GIS Map Object**

By clicking on the map object, you can adjust the **Properties tab** to select the source of the **Routes** that will be created during model execution either from the OSM server or PBF file, the criteria for route selection (fastest, shortest), the road type such as car/vehicle, bike or walking, and other properties. Select
shortest and OSM (open street maps) for the properties (see Figure 3)

![Map object properties](image)

Change the zoom and the position of the map in the view area to display the Midwest/Chicago area.

As the map object is interactive with the model in real time both at design and run time, you can double click the map object and activate the GIS provider’s map object and adjust the zoom level and pan to our region of interest (Figure 4). Next add points to the map representing airports.

Using the built-in search function from the GIS map provider, enter “airports” into the search window. The search will result in a list of airports found in the map area (Figure 4). Highlight the desired airports (left mouse click) and continue with ctrl+left mouse click (or select a range by using shift and left mouse click to highlight a continuous range). Select one of the highlighted
airports (right mouse click) and select **Convert to GIS point**. The selected points will be highlighted on the map. Remove the remaining points from the search window. The map object will appear like **Figure 5**.

Using the search function, type “**Elgin**” which will be the location of the manufacturing center. Select the city of Elgin from the search results and clear the rest.

**Figure 4: Searching for objects within the map object**
5. Collection of Agents

Since you will have a single manufacturer supporting a network of airports, you need to group the airport locations into a collection. To do this, select all airports tags from the map and select **Create a collection**. Be sure to **de-select the Elgin** location as this is NOT an airport but rather the manufacturing location. In the collection properties define the name **airportLocation**. Just as the name states, this will be a collection of airport coordinates.

It is a requirement to place airports into these coordinates. To do this, create a population of agents, which will be airport models. From the palette **Agent** drag into the editing area component **Agent**. A window opens “**Create and agent**”. The **Figure 6** shows the dialogue box for creating a new agent.
Figure 6: New Agent dialogue box

Select **Next** after each step as shown in *Figure 7 - 10*.
Figure 7: Specifying the agent name
Figure 8: Animation selection for the agent type
Figure 9: Agent Parameters dialog box
After pressing Finish, an icon will appear on the main object called “airports[..]” which is the agent population called airports[..]. There is now a need to relate airports[..] with coordinates of airports and with the number of airports, i.e. link with the collection airportLocation.

Referring to Figure 11, go into the population properties, by clicking airports[..] and input the following expression into the field Initial number of agents:

airportLocation.size()

This function returns the number of elements in a collection airportLocation (in this case 8). To link airports with their coordinates, in item Initial
location select **Position agents: in the node** and in the **Node** column enter the expression:

```
airportLocation.Get(index)
```

This function returns the address (indices) of the elements from the collection **airportLocation** (to those labels on the map).

![Properties of Airport agent](image)

**Figure 11: Properties of Airport agent**

Run the model, by pressing F5 key, to verify the data that we have. If everything was done correctly, you will see the following screen shown in **Figure 122**.
6. Create Manufacturing Agent

The model will need a manufacturing agent with a set of delivery trucks (resources) and logic to process orders from airports as well as methods to deliver and return trucks. Therefore, create the manufacturing agent and build the necessary logic.

Now create an agent type with the name `manufacturing` for placement of the business in Elgin. The agent is created in a similar manner, as shown in the Figure 6 through Figure 10, with some minor differences:

1. Instead of populations of agents only select a single agent.
2. Create a new agent type.
3. The name of the new type: `manufacturing`.
4. Animation to choose: 2D/plant.

Figure 12: Model execution with Airport Agents located within the Map Object
5. The parameters field is left blank.

In the main object, you will see an icon created for the new agent (factory). Now you need to link the manufacturing agent with the mark on the map. Open the manufacturing agent properties and in section initial location select Place agents: in the node and in the column Node access drop-down list, and select Elgin, as shown in Figure 13:

![Figure 13: Manufacturing Agent properties window](image)

Next, check what happens when you run the model.
Figure 14: Model execution with both Airport and Manufacturing Agents located properly within the Map Object

7. **Create Truck Agents**

From the problem text you know that delivery is made by trucks, to add them in the model, it is necessary to create a new agent population. This is done similarly, as in the *Figure 6 through Figure 10*, with some modifications:

1. Create a new **Population of agents**.
2. The name of the new type: **truck**.
3. Select animation: 2D: **Lorry**.
4. Specify the options according to **Figure 15**;
Use the “Client” parameter as a placeholder (or pointer) for which airport has created the order and is requesting this specific order’s spare parts. The “Client” parameter is declared as a type “Airport” so that the simulation can easily determine which airport has requested this specific order. Use a similar parameter in the “customer” in the Order agent and assign order’s customer to the truck’s client.

For further optimization, bring the number of trucks as a separate parameter. For this you want to pull out Parameter from the Agent palette. Set the following properties:

1. Name: numberTrucks
2. Type: int
3. Default value: 5

It is understandable that all trucks are owned by production (agent manufacturing), it is necessary to specify the starting location in the agent population trucks properties, such as in *Figure 16*:

![Figure 16: Truck Agent properties, starting location set to GIS Node Elgin](image)

From the problem, it is known that to obtain spare parts airports should fill out an order. To reflect this in the model create a new agent, as in the *Figure 6 through Figure 10*, with the following modifications:

1. Instead of populations of agents, choose Agent type only.
2. The name of the new type: Order.
3. In paragraph Animation select No;
4. Specify the options according to the Figure 17;

![Image of New agent step 4: Agent parameters]

*Figure 17: Parameters for the Order Agent type, with Customer property declaration*

The parameter “customer” is declared as a type “Airport”. This will be useful as the orders are created by airports and received by the manufacturers. Again, you can use this parameter in the order to assign the airport that has created the order to the delivery truck.

It is now necessary to show that trucks are resource of the production. To do this, first open Manufacturing agent from tab Projects (or Main). Then, add a
resource pool from the palette Process Modeling Library, that specifies a set of available resources, in this case - trucks. Next, you need to set the parameters the Resource Pool, according to the Figure 17:

![Truck agent properties window](image)

*Figure 18: Truck agent properties window*
The number of resources corresponds to the parameter numberTrucks value created earlier, it should be noted that this unit adds a new block, trucks to new resources (main.trucks).

Run the model to test the changes. If all is correct model would be consistent with the table in *Figure 19*.

*Figure 19: Model run with Airports, Manufacturing, and Truck Agents displayed*

The initial data entry can be considered completed. Initial data include airports (their coordinates), production (coordinates), trucks (as a resource of production), created an application for ordering spare parts (such as agent type order). We can change the scale of the truck representation if necessary.

Turn now to the logic of the work of airports and production.
8. **Airport States and Orders**

The logic of the work of airports, in this task, can be seen as successive transitions of the state of normal operation to standby state waiting for spare parts and back. To simulate this process it is necessary to take advantage of palette **Statechart**.

Select the airports[..] agent and open a palette **Statechart**. Any state chart should begin with **Statechart Entry Point**- it is a single point of entrance into the chart condition. It needs to be connected with the **State** element, which will be responsible for the normal operation of airports (the period of time when spare parts are not needed). Specify the name **normalWork**, the properties changes are not necessary. Create another state in the working area and name this condition **waitingDetails**. You can return to the properties for this element at a later time.

As stated in the problem, airports require spare parts at some frequency, in this case, 7 times per week. To do this, create a transition from the Normal state to the Waiting state and set the following properties:

- Triggered by: Rate
- Rate: 7 times a week.

As soon as the airport is in waiting for spare parts mode, an order has to be filled out and sent to the manufacturer. First, an order needs to be created and then sent to the manufacturer. You can accomplish this within the Waiting state’s properties entry action:

```java
Order order = new Order (this);
Send (order, main.manufacturing);
```
In the first line there is a new order with the parameter `this`, which indicates an airport, from which an order is received. The second line of code is needed to send - `send` function, which has two arguments: what to send (order) and to whom to send (on the production - agent `manufacturing`).

To ensure that the chart position fully reflects the logic operation of an airport, in the context of this task, there is a need for another transition from state `waitingDetails` to state `normalWork`. As known from the problem, the normal mode of operation is carried out when you receive a message "Delivered!". For this, the properties of this transition should be entered as seen in *Figure 21*. 

*Figure 20: Airport waiting state properties*
Figure 21: Transition from Waiting to Normal ... triggered by the message “Delivered!”

Figure 22: The completed state chart for the Agent Airport ... Normal working and Waiting for deliveries with transitions between each state

The airport can now send orders for spare parts. Turn to the processing and execution of the order.
9. **Manufacturing and Order Processing**

After receipt of the order, the production allocates a resource (truck) for the execution of this order. The truck is loaded with ordered spare parts, for which it requires 2-3 hours and then they are sent to the airport. Upon arrival at the airport, the trucks are unloaded for a period of two to three hours. After unloading is completed, the “Delivered!” message is sent and the truck returns to the production, and becomes a free resource.

Below, *Figure 23*, shows the design of the process, then each unit will be considered separately.

![Process Design](image)

*Figure 23: Processing modeling logic for the manufacturing agent ... indicating a resource pool of trucks, order processing and the parallel process steps for the trucks*

To realize the process shown above, open edit agent **manufacturing** and palette **Process Modeling Library**. Select the “Enter” block, which receives orders. Drag the block to the working area in the properties set.

- Agent Type: **Order**;
Next, received order enters the queue waiting for resources, to do this, drag the block **Seize**, which is responsible for resource capture and connect it with block **Enter**. Open the properties of block **Seize** and in item **Resource sets** click on image "+" - from the pop-up list, select **trucks**.

Before, as a resource will be acquired, it must be prepared, which in our case means packing. In a parallel process, the trucks will undergo the packing and delivery and eventual return process steps, outlined in the Resource Task Start and Resource Task End process.

Select **Resource task start** block and drag and drop from the palette **Process Modeling Library** and position below, as shown in *Figure 23*. In this unit the properties you will need to specify:

- o Unit Type: **truck**.
- o Start here: all the resources.

To pack a resource requires time. Drag and drop the unit **delay**, call it **packing** and connect with the **Resource task start**. Unit **Delay properties**, are shown in the *Figure 24*. 
The time delay is specified uniform distribution between two and three hours: uniform (2, 3).

Once a resource is loaded, it will be sent to the airport. Drag the item **Move to**, rename it to **toAirport** and connect as shown in Figure 23. In the properties it is necessary to specify the destination, as shown in Figure 25.
Resource (truck) is sent to the unit seize, now this resource should be transferred the information inside your order, so that the truck moves to the airport that created this order. To do this, go to the properties of the object seize and in paragraph Actions column On seize unit write the following expression:

\[(\text{Truck}) \text{ unit}) \cdot \text{client} = \text{agent.customer}\]

In this expression the value of parameter client, which is stored inside a resource truck, is assigned the value of the parameter customer agent Order.

As soon as the truck arrives at the airport, it must be unloaded. This process is modeled similar to the process of loading the truck, namely the delay, which in this case will be referred to as unpacking. The properties of this unit are shown in Figure 26.
When modeling logic operation of the airport it was pointed out that the transition from the standby waiting for spare parts to normal working state happens when you receive a message "Delivered!". This message is being sent, as soon as unloading is finished. Inside function **Send** which has two arguments (that to send, who to send) is used to send a notification.

After the order is finished the resource becomes free (block **release**, Figure 23), and the agent (**order**) is sent to the unit **Sink**, where it will be deleted. Since a
resource is free, it must be sent to the production, to do this, once again use unit **Move to**, in which to specify a destination agent **manufacturing**, which is located on the main agent (**Figure 23**).

Sub-process for the resource must be completed by the resource **Task end**, to ensure that the resource has returned to a common pool resources and was available for the new pick-up.

With this last step, the process of orders is ready, only thing left to do is make sure that all incoming orders were received at the unit **enter** processing.

All incoming messages are processed in a standard block **connections**, which by default, is within each agent. Open the properties of the unit **connections** inside the **manufacturing** agent (**Figure 27**):

![Figure 27](image)

**Figure 27**: The unit “connections” used to put the received order agent into the “Enter” block of the process logic
Figure 28: Communication properties of connections

In the field **On message received** the command `enter.take(msg);` is used. This command simply “takes” the message, which is the actual order and puts the order into the “Enter” block of the process logic (*Figure 23*). Message type is indicated earlier in agent **manufacturing** (message type is Order).

Model of the delivery of the parts to the airports is ready and you can now run the model.

10. **Run the Model**

Before you run the model, you must change the unit time range. To do this, open the tab **Projects**, highlight the project **AirportDelivery** and in unit properties **units model time units** specify the **days**. You can run the model by pressing F5. The result of executing the model is listed in *Figure 29*.
Figure 29: Delivery model execution

If everything was done correctly, trucks can be seen leaving the manufacturing facilities, driving to the airports and returning. Changing the scale of a map, you'll see that trucks are moving on real routes, which, just like the map, are imported from the network during the model execution.

To assess the occupancy of the company’s vehicle fleet, open the agent **manufacturing** when running the model. For this move to the display area of model, pressing and holding the right mouse button (pan).
Figure 30: Manufacturing agent during model execution

When opening the agent **manufacturing**, please note the unit **resource pool (trucks)**. The order speed processing is a fixed number of captured resources, on which you can assess the effectiveness of resources. The use of five trucks, the average asset utilization is approximately 45%, only between two to three of the five trucks are involved most of the time (*Figure 31*). In this model, an optimization experiment will be utilized to determine the ideal fleet size needed to maximize fleet utilization nearest 85% utilization.
11. **Optimization Experiment**

To optimize it is necessary to produce a new experiment. For this purpose, on the **projects tab**, bring up the context menu in the **Simulation: Main**, next to **Create/experiment**:  

![Call New Experiment Dialogue Box](image)

**Figure 32: Call New Experiment Dialogue Box**

Here select the **Experiment Type: optimization** and click Finish.

Go to the properties experiment and modify them, respectively, in **Figure 33**.
As the target parameter that can be changed is selected **numberTrucks** (number of trucks), which changes discretely from 1 to 5. The objective function for this optimization is defined as maximizing the number of trucks required to meet a specific target of fleet utilization. As seen in Figure 33, type the code `root.manufacturing.trucks.utilization()` in the objective parameter. Set the Requirements parameters for the same utilization reference, but to no more than 85% (less than or equal to). This will set a max utilization value of 85% to allow for a reasonable amount of maintenance (as an example) on the truck fleet.

Run the experiment for optimization, by selecting it from the tabs **palette**. Figure 34 shows an example of an experiment.
Figure 34: Optimization results

As can be seen from Figure 34, the best valid value for truck utilization is obtained at 3 trucks with a 72% utilization. Less than 3 trucks, the utilization is greater than 85% which is violation of the requirement function of less than or equal to 85% utilization.

By selecting “copy” from the optimization window, the optimal solution will be copied to the clipboard for importing into the simulation.

Running the simulation with the results obtained from the optimization experiment will produce a truck utilization of between 70 to 72% utilization based on a fleet size of 3 trucks.