

# Basic Models

**AnyLogic**

**Author: Alejandro Rodriguez Grau**

**Tutor: Prof. Dr. Angel A. Juan**

# Full Course Index

---

- **Introduction to AnyLogic**
- **Basic models**
- **Heterogeneous model**
- **Advanced model**

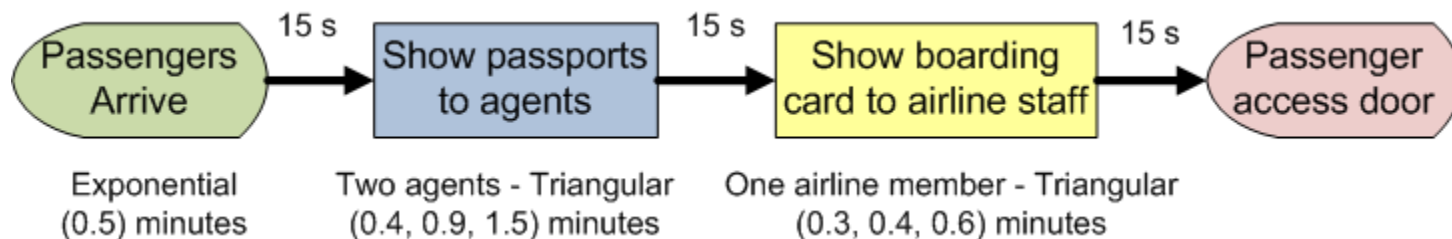
# Basic Models

---

- **Exercise 1**
  - **Access Control Point**
    - Overview of the problem
    - Create the Model
    - Run and Results
- **Exercise 2**
  - **Check-in Point**
    - Overview of the problem
    - Create the Model
    - Run and Results

# 1. A simple access control point

- Process overview: Passengers **arrive** to a passport control point. They **wait** in line and **show their passports** to any of two agents. Then, passengers **move** to the **boarding card** control. After validating their cards, they can **proceed** to the boarding door.
- Assumptions: (a) Passengers **interarrival times** follow an Exponential(0.5) minutes; (b) the time an **agent** takes to interact with the passenger follows a Triangular(0.4, 0.9, 1.5) minutes; (c) the time the **airline staff member** takes to interact with the passenger follows a Triangular (0.3, 0.4, 0.6) minutes; (d) the **travel time** between each process is 15 seconds; (e) passengers will wait in a **single waiting line** on a FIFO basis; (f) there is no limit to the **length** of the waiting lines.



# 2. Create a new Model, Add Agent Type

New Model

Create a new model

Model name: BasicModel1

Location: C:\Users\Alex\Models Browse...

Java package: basicmodel1

Model time units: minutes

The following model will be created:

C:\Users\Alex\Models\BasicModel1\BasicModel1.alp

Finish Cancel

This model will work best if each step represents 1 minute inside the simulation.

Drag & Drop an *Agent Type* from the *Agent Palette*.

Add Parameters and Variables on new Agent Type:  
**Passenger**

Main Passenger

parPathTime varNumInSys varTimeInSys

Properties Passenger - Agent Type

Name: Passenger  Ignore

Parameters preview

parPathTime:

# 3.Parameters & Variables for optional use

We could define a parameter to fix the value of the time in a path.

The screenshot displays a modeling software interface with a statechart diagram on the left and a parameter configuration panel on the right. The statechart diagram, titled 'Main' and 'Passenger', shows a state 'connections' with a transition arrow pointing to a state 'parPathTime'. Below 'parPathTime' are two variables: 'varNumInSys' and 'varTimeInSys'. The parameter configuration panel, titled 'parPathTime - Parameter', shows the following settings: Name: parPathTime, Visible: yes, Type: double, Default value: 15, System dynamics array: unchecked, Value editor: Label: parPathTime, Control type: Text. Red arrows point from the text boxes to the parameter name in the diagram, the parameter name in the configuration panel, and the default value '15' in the configuration panel.

But in our case, we will use connectors into a *Delay* block, where *Entities* will wait those 15 virtual seconds, before proceeding to the next step.

Our Variables will be used, as in the first model, to obtain and print results. So for now, let's go back to the *Main* tab, and create the proposed model.

# 4. Implement your Model (1/2)

Drag & Drop the *Source*, *Services (2)* and *Sink*, change their names accordingly and set up the defined parameters on the first slide of this exercise

The screenshot displays a simulation software interface with a main workspace and a properties panel. The workspace shows a flow diagram with four components: 'Arrival' (a blue circle with a plus sign and arrow), 'Agents' (a blue square with a clock icon), 'Staff' (a blue square with a clock icon), and 'AccessDoor' (a blue circle with an 'X'). These components are connected by lines, and the entire flow is enclosed in a red rounded rectangle. A red arrow points from the text box above to the 'Arrival' component. The properties panel on the right is titled 'Arrival - Source' and contains the following settings:

- Name: Arrival  Show name  Ignore
- Arrivals defined by: Interarrival time
- Interarrival time: exponential( 0.5 ) minutes
- First arrival occurs: After timeout
- Set agent parameters from DB:
- Multiple agents per arrival:
- Limited number of arrivals:

# 4. Implement your Model (2/2)

Agents - Service

Name: Agents  Show name  Ignore

Seize:  (alternative) resource sets  
 units of the same pool

Resource sets (alternatives):

Queue capacity: 100

Maximum queue capacity:

Delay time: triangular (0.4, 0.9, 1.5) minutes

Send seized resources:

Agent location (queue):

Agent location (delay):

Staff - Service

Name: Staff  Show name  Ignore

Seize:  (alternative) resource sets  
 units of the same pool

Resource sets (alternatives):

Queue capacity: 100

Maximum queue capacity:

Delay time: triangular (0.3, 0.4, 0.6) minutes

Send seized resources:

Agent location (queue):

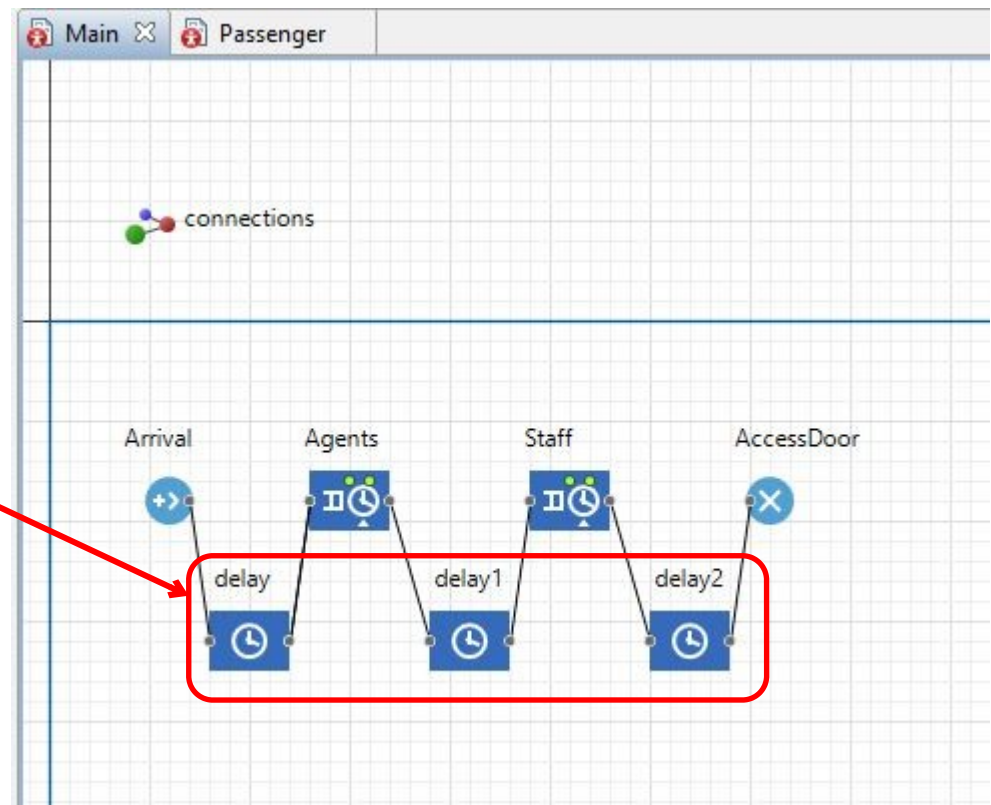
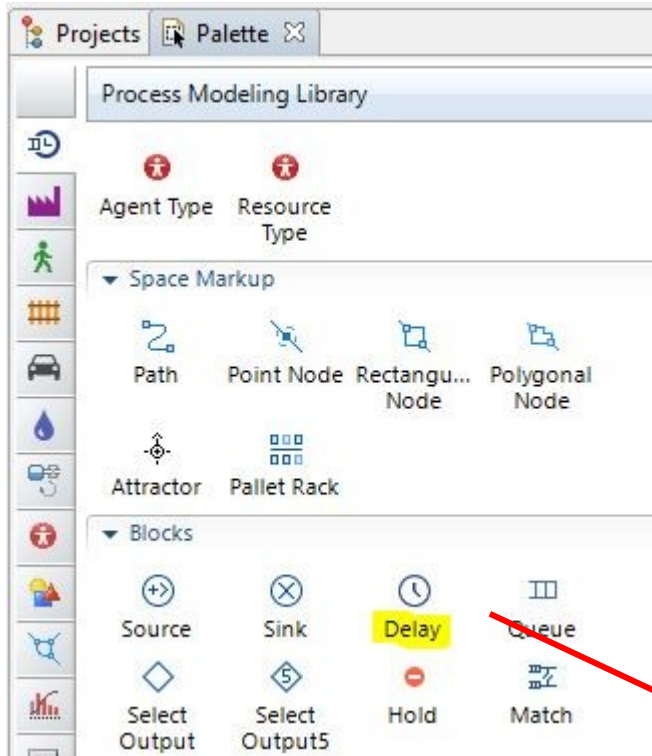
Agent location (delay):



# 5. Add Delay (1/2)

Now we already have a simulation to run, entities have an arrival rate, processing time on the server and a sink to close the cycle.

Still, entities are **instantly** going from the “out” nodes to the “in” nodes. We must add a *Path* or a *Delay*, where we can specify the time that it takes for a passenger to go from one place to another (in this case, a fixed time).



# 5. Add Delay (2/2)

The screenshot displays a simulation environment with a process flow diagram on the left and a properties window on the right. The process flow diagram shows a sequence of steps: Arrival, Agents, Staff, and AccessDoor. Each step is connected to the next by a line, and a delay block is placed on each line. The delay block is highlighted with a yellow box. The properties window, titled 'delay - Delay', shows the following settings:

- Name: delay
- Type: Specified time (selected)
- Delay time: 15 seconds
- Capacity: 100
- Maximum capacity: (empty)
- Agent location: (empty)

A red arrow points from the 'Capacity' field in the properties window to the 'delay' block in the process flow diagram.

Once defined the waiting time, we must set a high enough capacity, because if we leave only 1 (by default), no other entity will be able to do that path until the one in it reaches the exit.  
Do this for every Delay block

# 6. Use *Parameters* in Main (1/2)

Now that we know how to input data, lets learn how to substitute a number that appears several times and can be compressed into a *Parameter* inside the "Main" *Agent* screen.

The screenshot displays the NetLogo software interface. On the left is the 'Agent' palette with various components. The 'Main' agent screen is open, showing a flowchart with nodes for 'Arrival', 'Agents', 'Staff', and 'AccessDoor', connected by 'delay', 'delay1', and 'delay2' blocks. A 'parPath' parameter block is also visible. On the right, the 'Properties' window for the 'parPath' parameter is shown, with the 'Default value' field set to '15'. Red boxes highlight the 'Parameter' icon in the palette, the 'parPath' block in the agent screen, and the '15' value in the properties window.

Agent Components:

- Agent
- Parameter
- Event
- Dynamic Event
- Variable
- Collection
- Function
- Table Function
- Custom Distribution
- Schedule
- Port
- Connector

Main Agent Screen:

- connections
- Arrival
- Agents
- Staff
- AccessDoor
- delay
- delay1
- delay2
- parPath

Properties - parPath - Parameter:

- Name: parPath
- Ignore:
- Visible:  yes
- Type: double
- Default value: 15
- System dynamics array:
- Value editor: Label: parPath, Control type: Text

# 6. Use *Parameters* in Main (2/2)

Use that parameter instead of the numbers in each Delay block.

The screenshot displays a simulation environment with a process flow diagram on the left and a properties panel on the right. The diagram shows a sequence of nodes: Arrival, Agents, Staff, and AccessDoor, connected by arrows. A path labeled 'parPath' is highlighted in yellow, starting from the Arrival node and passing through three delay blocks labeled 'delay', 'delay1', and 'delay2'. The 'delay' block is highlighted with a red box. The properties panel for the 'delay - Delay' block shows the following settings:

- Name: delay
- Show name:  Ignore:
- Type:  Specified time  Until stopDelay() is cal
- Delay time: parPath
- Capacity: 100
- Maximum capacity:
- Agent location:
- Advanced section:
  - Agent type: Main
  - Single agent:  Population of agents:
  - Model/library: Process Modeling

Remember to change the *Agent Type* on the *Delay* blocks so the program can find its way to the parameter!

# 7. Add Results

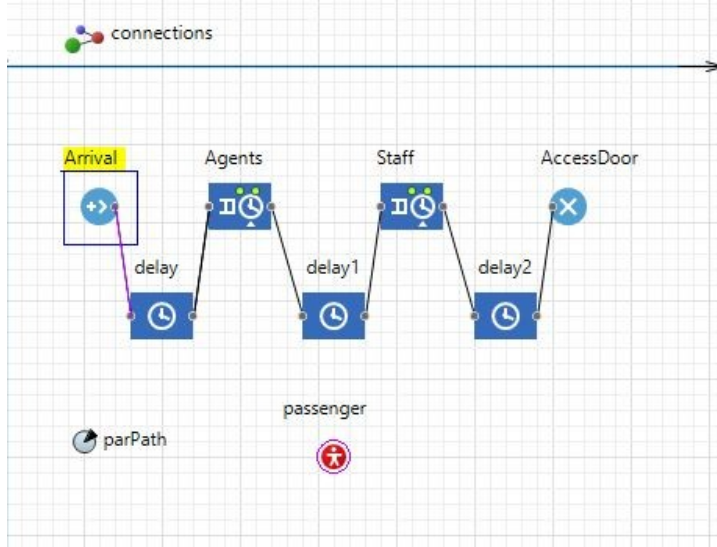
Time to revisit the *Passenger Agent*.

Reorder parameters and variables according to what is needed, some can go in Main, others must go in the Agent in question.

The screenshot displays a software development environment with two main windows. On the left, a project browser shows a 'Main' window and a 'Passenger' window, with 'Passenger' selected and highlighted by a red box. Below this, a diagram shows a 'connections' node with three colored spheres (blue, green, red) and a horizontal arrow pointing right. Below the arrow, three variables are listed: 'varInTime', 'varOutTime', and 'varTimeInSys', each with a 'V' icon and highlighted in yellow. On the right, a 'Properties' window for 'varOutTime - Variable' is open. It shows the following settings: Name: 'varOutTime', 'Show r' checked, 'Ignore' unchecked, 'Visible' set to 'yes', 'Type' set to 'double', 'Initial value' empty, 'Access' set to 'public', 'Constant' unchecked, 'Save in snapshot' unchecked, and 'System dynamics units' empty. A red arrow points downwards from the right side of the Properties window.

Notice that it's no longer as seen in slide number 5, this is not unusual, as when you program, you sometimes find better ways or improvements for the system performance along the way.

# 8. Show Results on console (1/2)



Multiple agents per arrival:

Limited number of arrivals:

Location of arrival:

▼ Agent

New agent:

Change dimensions:

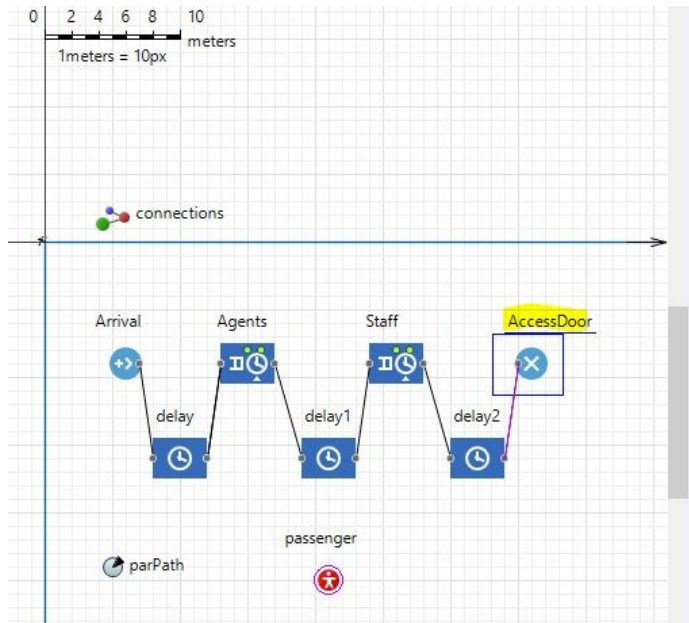
► Advanced

▼ Actions

On before arrival:

On at exit:

On exit:



▼ Actions

On enter:

▼ Advanced

Agent type:

Single agent  Population of agents

Model/library:

Visible:  yes

Visible on upper agent

Log to database

[Turn on model execution logging](#)

► Description

# 8. Show Results on console (2/2)

The screenshot displays the AnyLogic simulation environment. The main workspace shows a process flow diagram with the following components:

- Arrival:** A blue circle with a right-pointing arrow and the number 144.
- Agents:** A blue square with a play button icon, the number 0, and the number 144.
- Staff:** A blue square with a play button icon, the number 0, and the number 144.
- AccessDoor:** A blue circle with an 'X' icon and the number 144.
- Delay Blocks:** Three blue squares with a clock icon and the number 0, labeled 'delay', 'delay1', and 'delay2'.
- parPath:** A blue circle with a play button icon and the number 15.
- passenger root.passenger:** A red circle with a person icon.

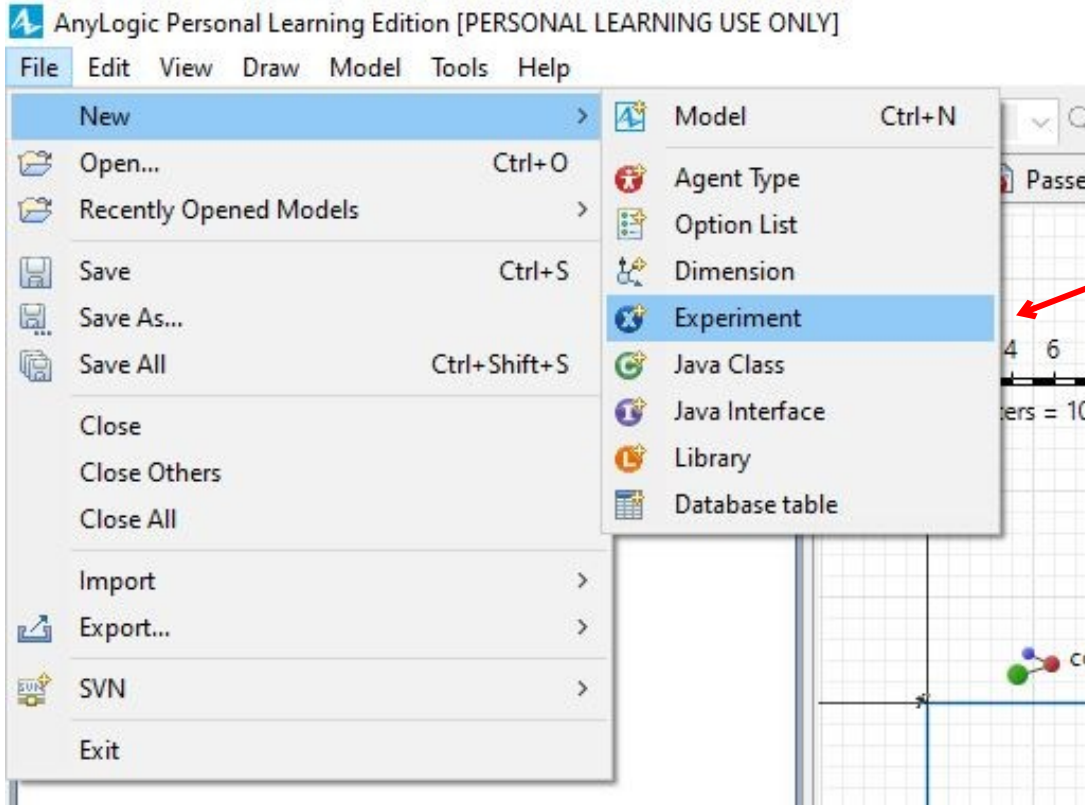
Connections are shown between these elements, with the number 144 indicating the count of agents at each stage. A status message 'connections not connected' is visible in the top left.

The console window on the right shows the simulation running for 336.95 minutes on 06/09/2021 at 05:36. The console displays a list of events with timestamps:

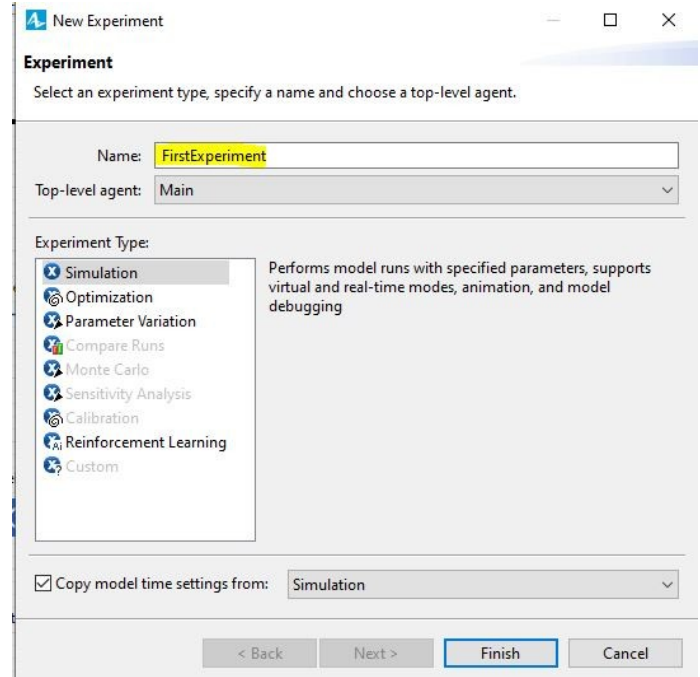
```
1. 9204761899810592
2. 218348326574926
2. 5193490618412966
1. 0226524881354067
0. 05014271921083946
0. 6755903778035872
0. 8658177515833927
0. 276179643608657
0. 10919200909643223
0. 5945893371349484
0. 4694161520069997
1. 0171405176538997
1. 9390003530178888
2. 0618761632833866
0. 2956517548591364
2. 1446116379826208
1. 5520289294368013
1. 6166322595809675
2. 007543618673594
2. 0088979833864187
0. 8727695474775032
0. 3108850768515481
0. 9061258671462156
0. 6878652381320194
2. 0260927919544542
2. 1012566473501693
```

Performance metrics at the bottom of the console window are: 27.00 sec/sec, EPS: 2, FPS: 39, Step: 2,304. The simulation is running for 21.16 seconds, which is 7% of the total 512M simulation time.

# 9. Add other simulations



Add an Experiment to your model workspace.

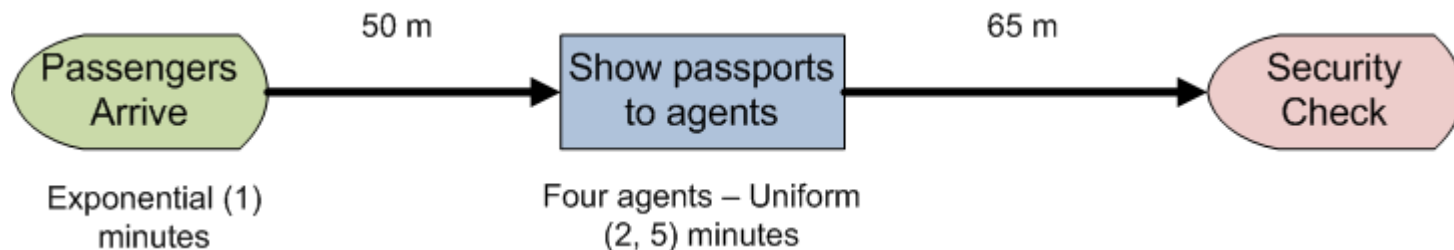


Adding a new experiment is basically adding a new mode of simulation for the program. We can change things as randomness, repetition, speed, or others present in the properties when selecting the new simulation on *Projects*.



# 10. A simple check in point

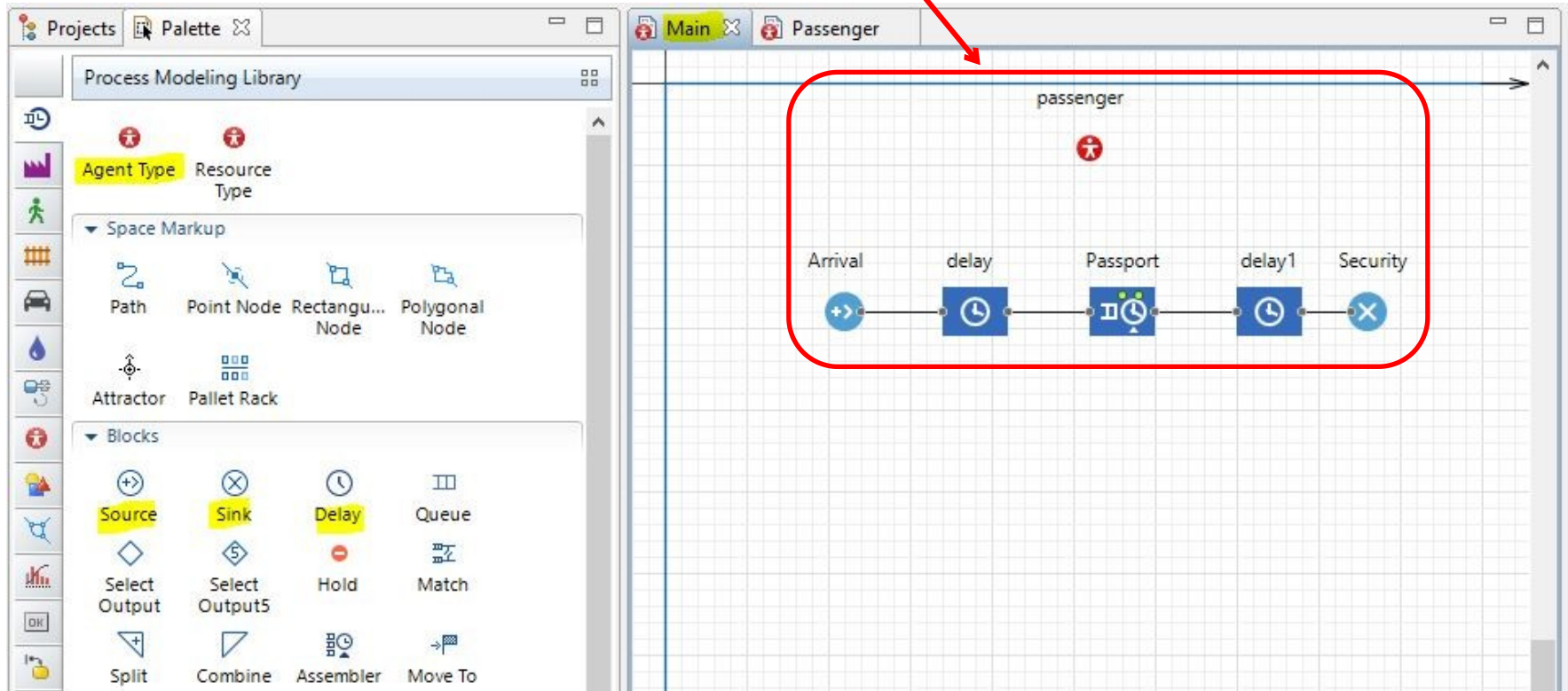
- Process overview: People arrive to an airport and go through the **check in** process to get their tickets. After check-in the passengers proceed to the security check point.
- Assumptions: (a) Passengers **interarrival times** follow an  $\text{Exp}(1)$  min.; (b) passengers **walk rate** follows a  $\text{Uniform}(2, 4)$  km/h; (c) passengers **travel** 50 m from the terminal entrance to the check in station; (d) following the check in process they must **walk** 65 m to the security check point; (e) the **check-in station** has 4 people assigned to process customers, who wait in a FIFO single line; (f) the **check-in process** follows a  $\text{Uniform}(2, 5)$  to completion; (g) the simulation model needs to be **run** for 24h.



# 11. Create the model

Follow the same steps as before, Drag & Drop objects into the model, connect them via a connector.

We will use Delay blocks to account for the time the passenger stays on the path.



# 12. Agent Passenger

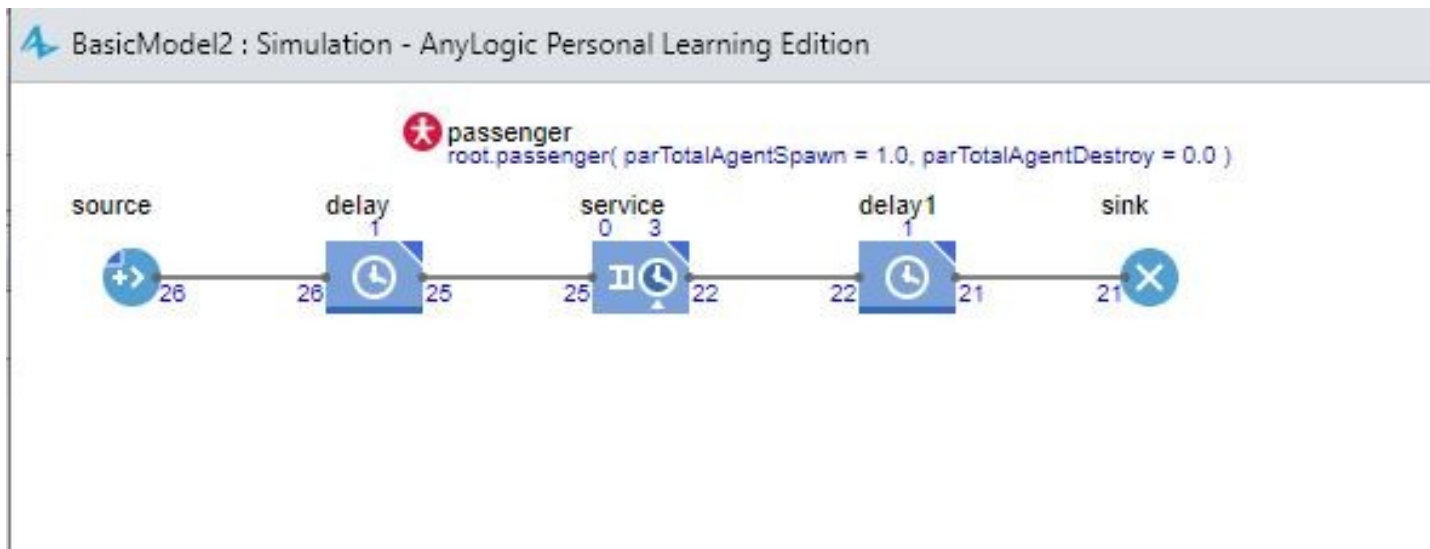
We create our Agent called Passenger, and we add variables and parameters that we need to show results.

Don't forget the metrics you are using! (This example shows the number for a simulation which each step equals 1 second)

The screenshot displays the software interface for defining a variable. On the left, a grid shows several variables: `parTotalAgentSpawn`, `parTotalAgentDestroy`, `varSpeed`, `varPassportRate`, `varTimeIn`, `varTimeOut`, and `varMeanTime`. A red rounded rectangle highlights the `varSpeed` variable. On the right, the 'Properties' panel for `varSpeed - Variable` is shown. The 'Name' is `varSpeed`, 'Visible' is checked, and 'Type' is `double`. The 'Initial value' is set to `uniform(0.5555,1.11111)`, which is highlighted in yellow. A red arrow points from the text box above to this initial value. The 'Advanced' section shows 'Access' is `public`, and other options like 'Constant', 'Save in snapshot', and 'System dynamics units' are unchecked.

# 13. First simulation

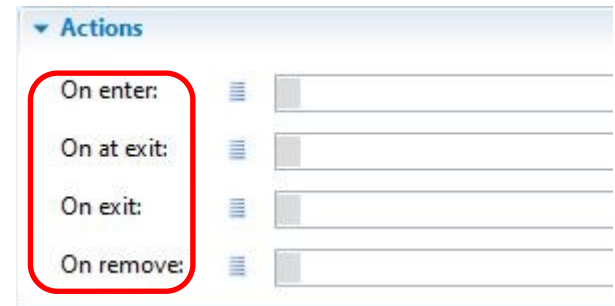
We can see how the simulation behaves, and we must check that it is not doing anything funny, like agents instantly skipping a block, travelling too fast, or not generating enough entities at the source.



With the simulation running we can set new parameters/variables to get data from the simulation on console.

# 14. Adding results

We must search for the Agent Actions on each block and use this spaces to order the program to produce results, it will be very helpful if you look first the example model, so you can see how this coding part looks like.



BasicModel2 : Simulation - AnyLogic Personal Learning Edition

connections not connected

passenger  
root.passenger(parTotalAgentSpawn = 321.0, parTotalAgentDestroy = 314.0)

source 321 delay 321 service 320 delay1 317 sink 314

18,958.23 sec 06/13/2021 05:15

WARNING [ 156.798] root.service.  
9.766121183451162  
11.694712062518875  
14.949734893090977  
17.656036259845905  
7.990439558540743  
119.17781575099127  
141.01291289368294  
179.8878383476923  
223.71033916018075  
9.55512885602161  
22.784261294650832  
10.888964438680205  
39.010371688907526  
63.815925394855185

3.61 min/sec EPS: 59 FPS: 40 Step: 3,182  
Running: 101.53 sec 10% of 512M