Bank Office Model

AnyLogic provides the Process Modeling Library, a discrete-event simulation library containing objects you can use to rapidly simulate complex discrete-events systems like:

- Manufacturing processes with detailed shop floor layout
- Simple and complex service systems (e.g. banks, airports, etc.)
- Business processes with activity based costing
- Logistics and supply chain models

The Process Modeling Library allows you to create flexible models, collect basic and advanced statistics, and effectively visualize the process you are modeling to validate and present your model.

In this tutorial we will create a simple service system of a bank department, consisting of an automatic teller machine and teller lines. ATM provides people with a quick self-service for cash. More complex transactions, e.g. paying bills, are completed by tellers, allowing customers more time without inconveniencing those customers looking for quick cash.

Note that there are several reference models available representing the milestones of the editing. You can use them if you experience any difficulties creating a model and you would like to compare your model with the reference file. The links to the reference models are given at the end of some steps. Just click the link to open the corresponding model.

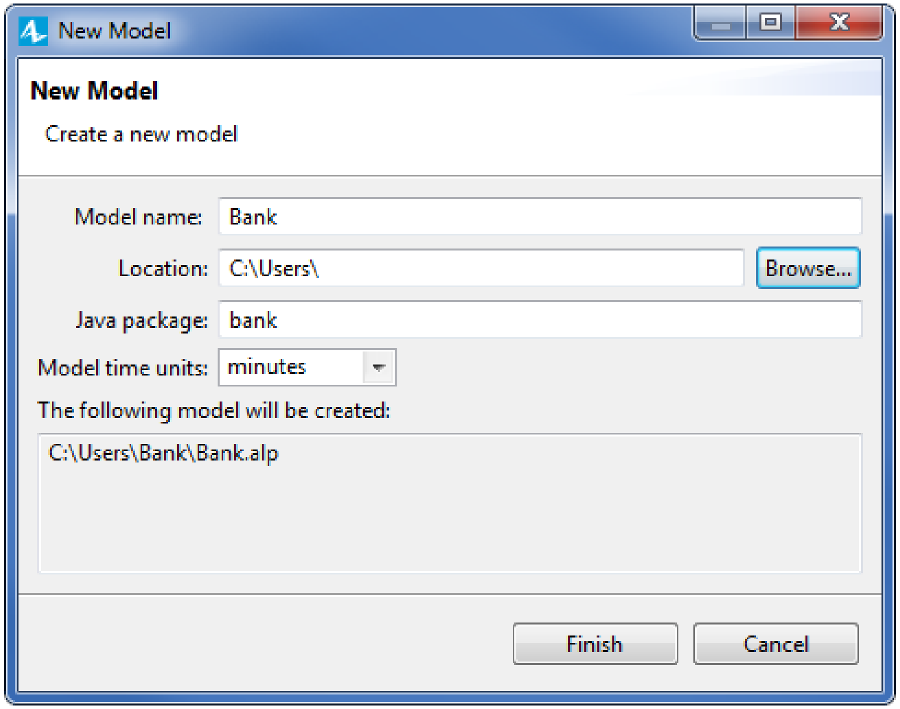
» Phase 1. Creating a simple model

Phase 1. Creating a Simple Model

First, we will create the simplest queueing model simulating how customers are serviced at the ATM.

Create a new model

1. Click the **New** toolbar button. The **New Model** dialog box is displayed.
2. Specify the name of the model. Type `Bank` In the **Model name** edit box.



New Model

Create a new model

Model name: Bank

Location: C:\Users\

Java package: bank

Model time units: minutes

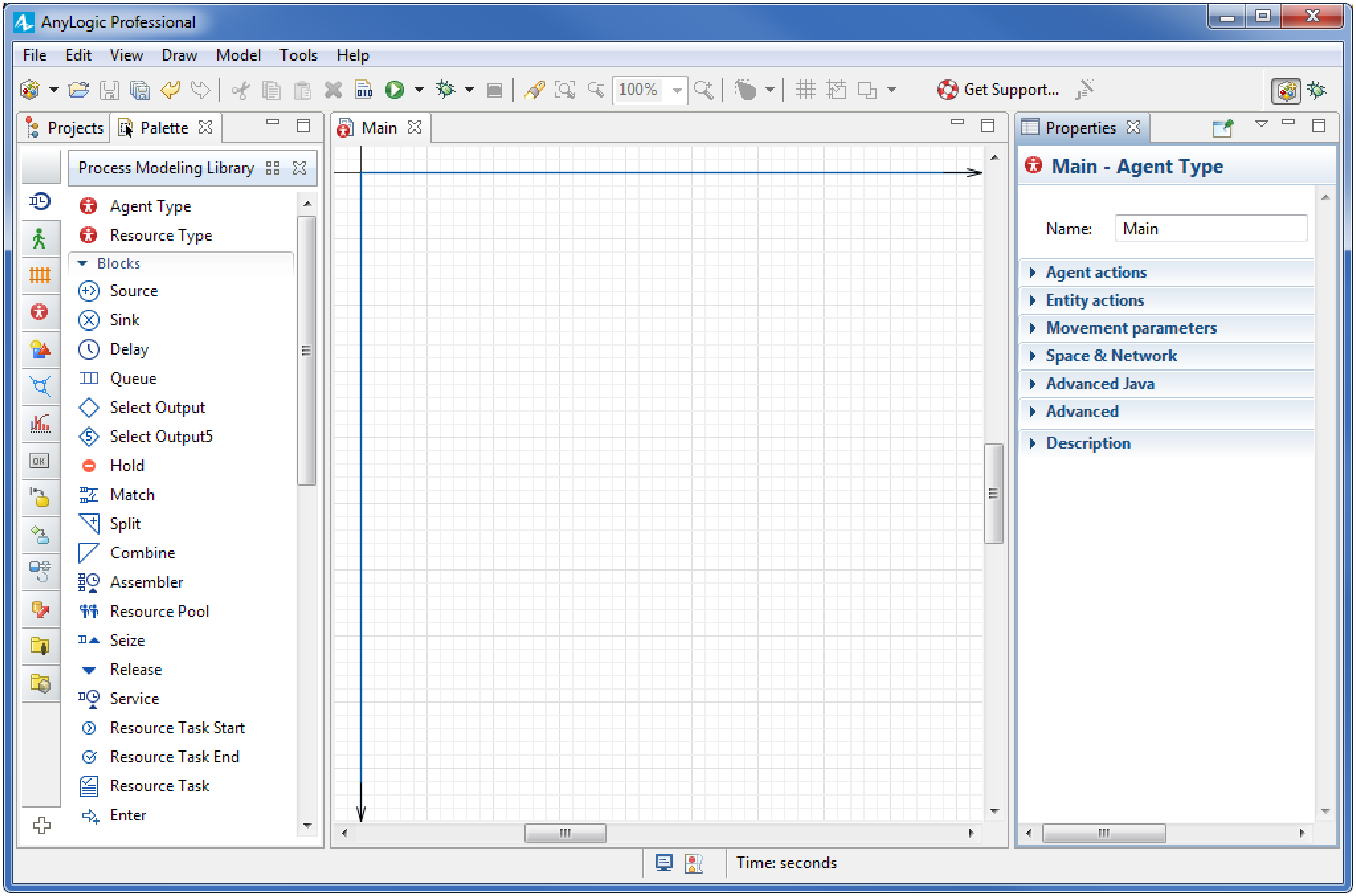
The following model will be created:

C:\Users\Bank\Bank.alp

3. Specify the location where you want to store your model files. Browse for the existing folder using the **Browse** button, or type the name of the folder you want to create in the **Location** edit box.
4. Select **minutes** as the Model time units.
5. Click **Finish** to complete the process.

New model is created. It already has one agent type called *Main* and experiment called *Simulation*. Agents are the main building blocks of AnyLogic model. In our case *Main* agent will serve as the place where will define all the logic of the model: here we will put the process flowchart, and define some simple animation.

In the center of the workspace you will see the graphical editor. It shows the diagram of the *Main* type.



To the left of the graphical editor you can see the **Projects** view and the **Palette** view sharing the same area. The **Projects** view provides access to AnyLogic models currently opened in the workspace. The workspace tree provides easy navigation throughout the models. The **Palette** view contains all graphical elements you can add onto the graphical editor of your agent just by drag'n'drop. Model elements are grouped by categories in a number of palettes.

On the right side of the workspace you can see the **Properties** view. The **Properties** view is used to view and modify the properties of a currently selected model element(s). When you select something – e.g., in the **Projects** view or in the graphical editor – the **Properties** view displays the properties of the selection.

Now we can start developing the model.

Creating the flowchart defining the process

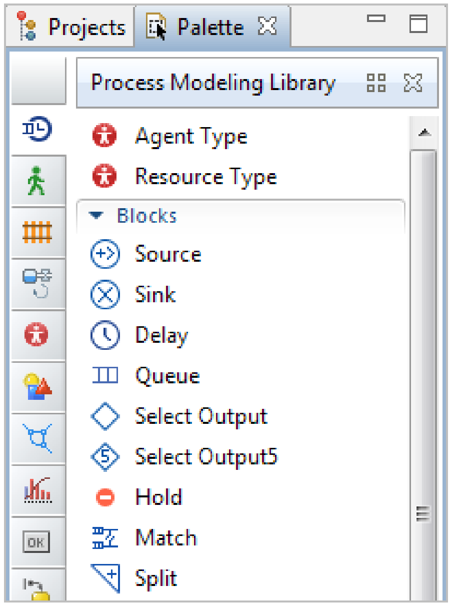
Now we will define the process with a flowchart composed from **Process Modeling Library** blocks.

Each block here defines some operation that will be performed with agents passing through this block.

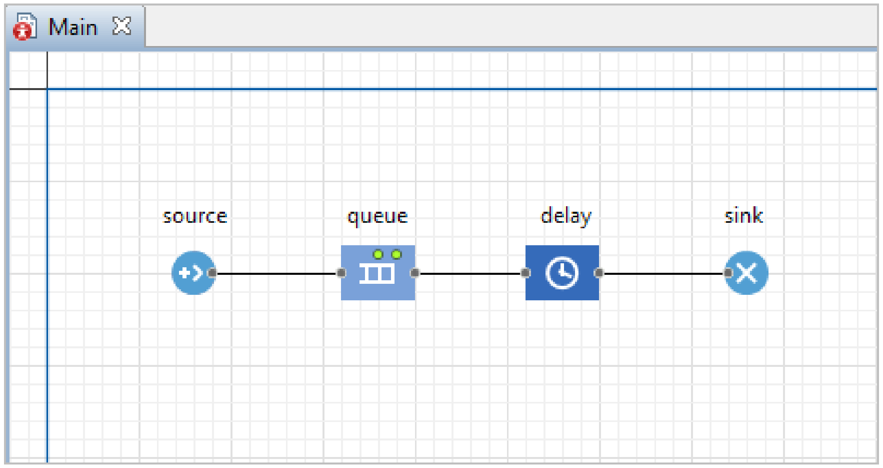
In AnyLogic you create flowcharts by adding the blocks from the library palette to the graphical diagram, connecting blocks together, and tuning the parameters of the blocks.

Create the model flowchart

1. After you create the model, AnyLogic will switch to the **Palette** view and display the **Process Modeling Library** palette:



2. Add **Process Modeling Library** blocks on the diagram and connect them as shown in the figure below. To add a flowchart block on the diagram, drag the required element from the palette into the graphical editor.



2. While dragging the blocks, and placing them close to each other, you may see the lines connecting the blocks appear. Please note that these connectors should connect only the ports lying on right and left borders of the block icons.

The given flowchart models the simplest queuing system, consisting of a source of agents, delay (and a queue before this delay) and final sink object.

Let's say a pair of words about these flowchart blocks.

Source block generates agents. It is usually used as a starting point of the process flow. In our example, it models customer arrival.

Queue block models queues. In this model it simulates a queue of customers waiting for the moment they can start accessing ATM services.

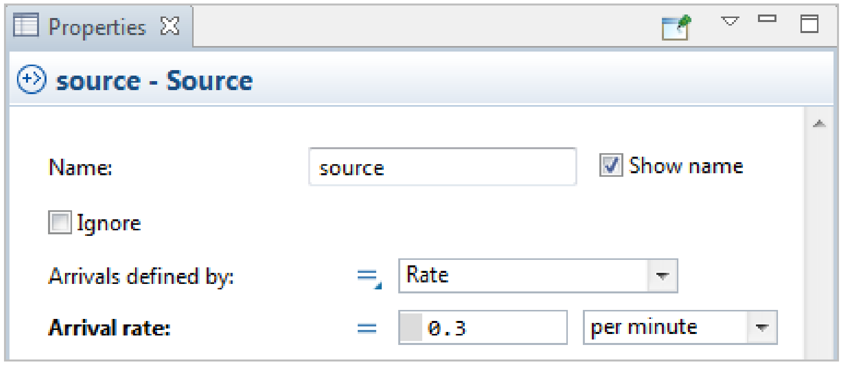
Delay here simulates the delay associated with the service at ATM.

Sink block indicates the end of the flowchart and discards the incoming agents.

Please refer to *Process Modeling Library Reference Guide* for the detailed information about all **Process Modeling Library blocks**.

Configure the flowchart blocks

1. To modify properties of some model element, first select it by clicking on it in the graphical editor or in the **Projects** view. This opens the properties of this element in the **Properties** view.
2. Select *source* block. In the **Properties** view, specify how often customers arrive. Type 0.3 and select *per minute* for **Arrival rate**.



Properties

source - Source

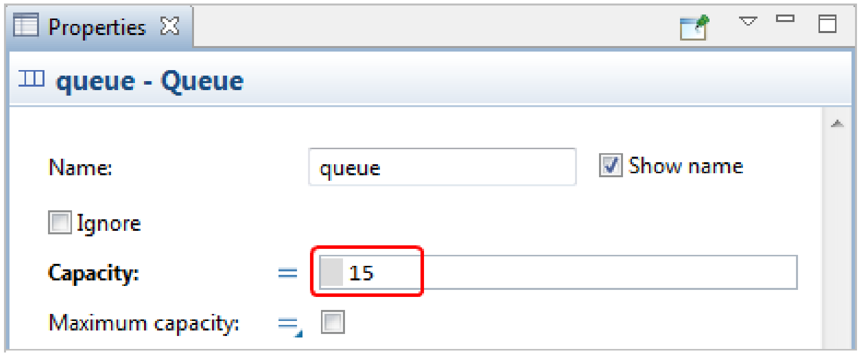
Name: source ☑ Show name

□ Ignore

Arrivals defined by: Rate

Arrival rate: 0.3 per minute

3. Modify the properties of the *queue*. Set queue capacity to 15 agents. At most 15 customers will wait in a queue.



Properties

queue - Queue

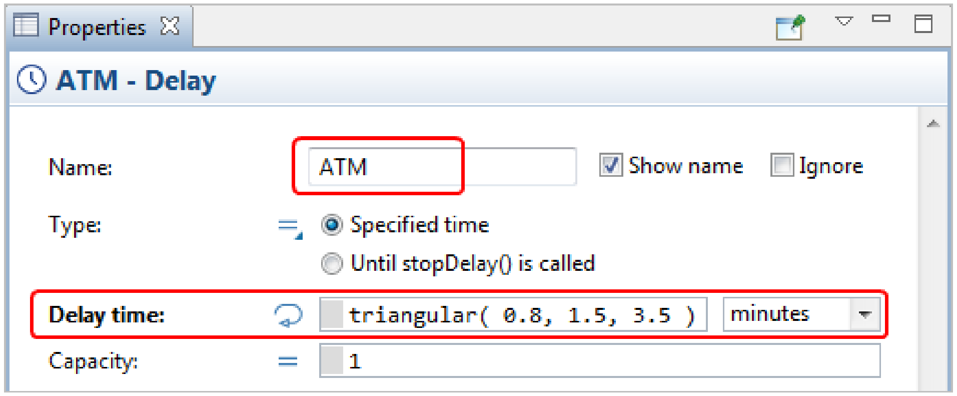
Name: queue ☑ Show name

□ Ignore

Capacity: 15

Maximum capacity: □

4. Modify the properties of the *delay*. Name the object *ATM*. Specify the processing time. Assume that processing time is triangularly distributed with mean value of *1.5*, min of *0.8* and max value of *3.5 minutes*.



Properties

ATM - Delay

Name: ATM ☑ Show name □ Ignore

Type: ● Specified time
○ Until stopDelay() is called

Delay time: `triangular( 0.8, 1.5, 3.5 )` minutes

Capacity: 1

`triangular()` function is the standard AnyLogic random number generator. AnyLogic provides also other random number distributions, like normal, Poisson, exponential, etc. Please refer to *AnyLogic Help* for the description of all the random number generators (see *AnyLogic Functions* topic).

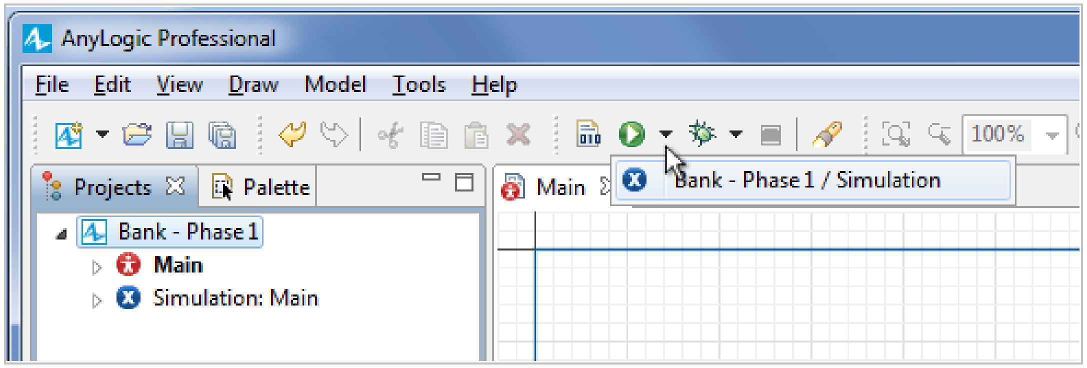
Running the model

Now we have finished modeling the simplest queuing system and we are ready to run the resulting model. First, build your model by clicking the **Build Model** toolbar button. If there are some errors in your model, the building fails and the **Problems** view appears listing all the errors found in your model. Double-click an error in the list to open the location of the error and fix it.

After the model is successfully built, you can start it. Running the simulation, you automatically bring the current model up to date.

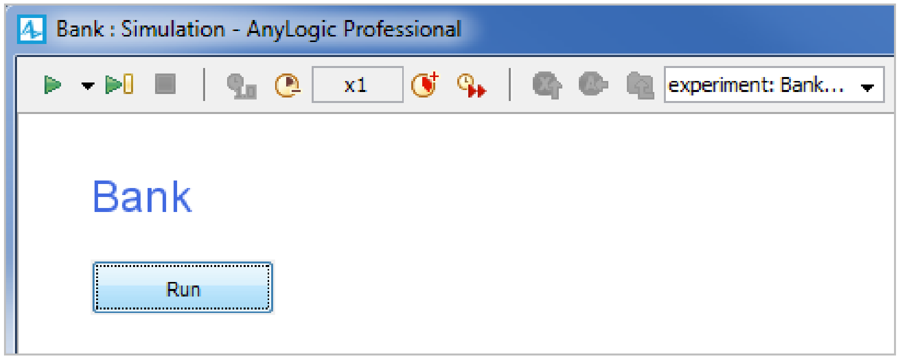
Start the model

1. Click the **Run** toolbar button and choose the experiment you want to run from the drop-down list. Your simulation experiment is called `Bank/Simulation`.

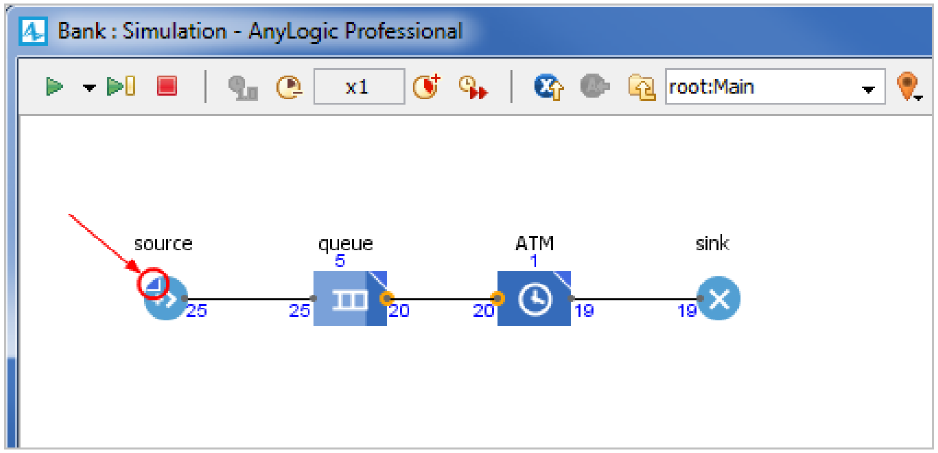


In the case this model is the only one opened in the workspace at the moment you will be prompted to run this particular experiment. Later on this button will start the previously run experiment. To run any other experiment, right-click (Mac OS: Ctrl+click) the experiment in the **Projects** view and choose **Run** from the context menu.

Having started the model, you will see the presentation window. It displays the presentation designed for your simulation experiment.

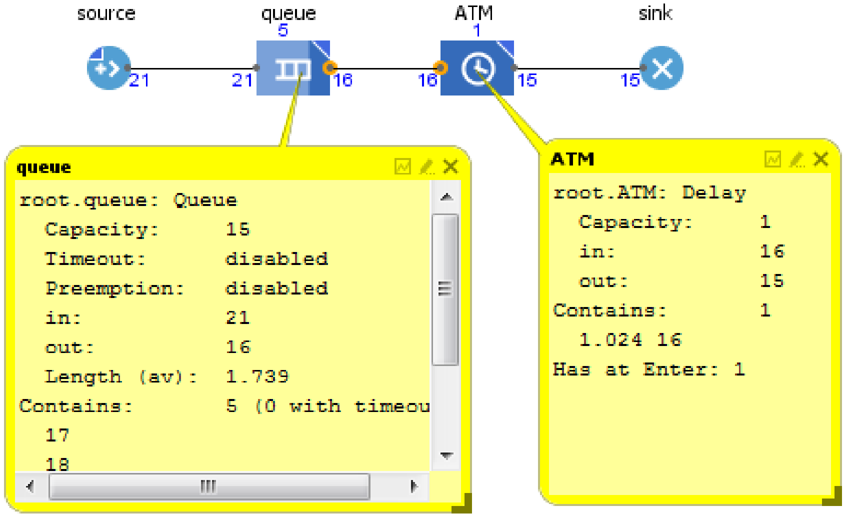


Click the button to **Run** and observe the process dynamics. You will see animated flowchart. Each model created with Process Modeling Library instantly has animated flowchart where you can see detailed current object status, for example queue size, number of agents left and so on – completely in graphics!



If needed, adjust the execution speed to your needs using **Slow down** and **Speed up** toolbar buttons.

You can inspect flowchart objects to get the detailed information on their current state. Click on the object to open its inspect window. Inspect window show statistics on the object, e.g. **Queue** object's inspect shows the queue capacity, the number of agents passed through either port of the object and also whether the timeout option is enabled for this queue. *Contains* string displays the number of agents currently being in the object along with IDs of these agents.



```
queue
root.queue: Queue
  Capacity:      15
  Timeout:       disabled
  Preemption:    disabled
  in:            21
  out:           16
  Length (av):   1.739
Contains:        5 (0 with timeou
  17
  18
```

```
ATM
root.ATM: Delay
  Capacity:      1
  in:            16
  out:           15
Contains:        1
  1.024 16
Has at Enter: 1
```

Reference model: Bank - Phase 1

Bank Office model

Phase 2, Creating model animation

Phase 2. Creating Model Animation

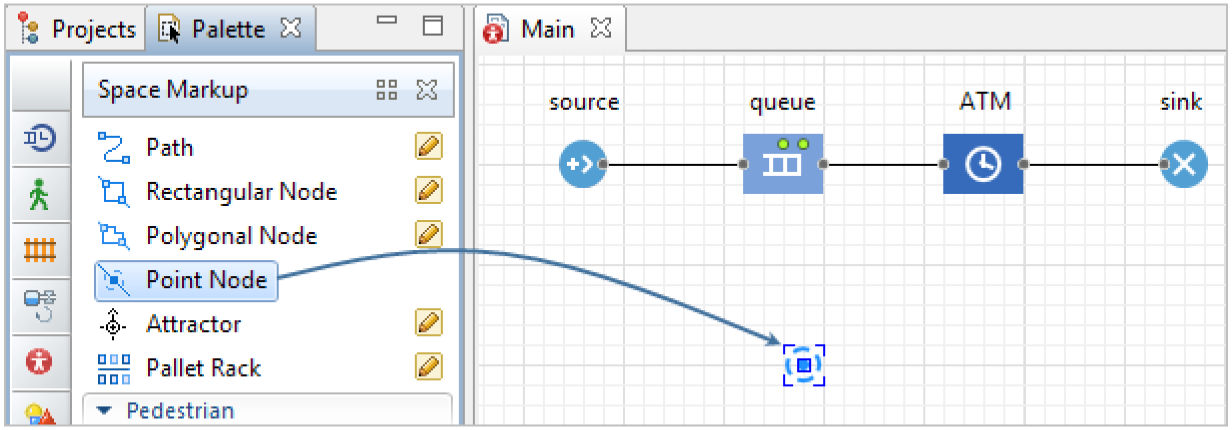
Although the flowchart is animated, you may want to see the actual bank department layout animated. That is also possible! For each model you can create an animation to visually represent your model. You can create any animation you want. Now we will draw the layout consisting of the ATM and a queue. Then we will animate clients standing in the queue and using the ATM. We also want to visualize the current status of the ATM.

Now we will draw the layout of our bank. You draw the layout on the same diagram where you draw a flowchart. However, if you have existing image of the layout, you can simply import this picture as the bank layout instead of drawing it by yourself.

Adding space markup shapes

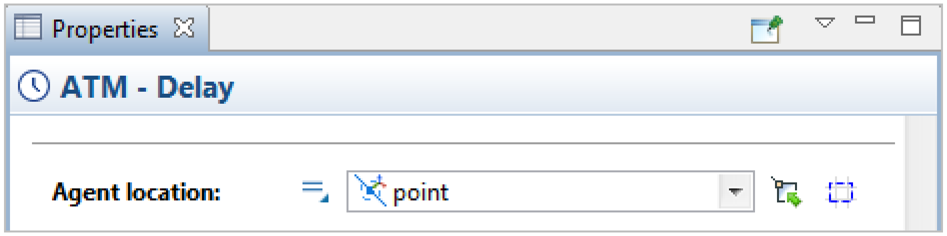
Set up space markup for the ATM

1. Draw the ATM as a point node. First, open the **Space Markup** palette in the **Palette** view.
2. Drag the element **Point node** from the **Space Markup** palette into the graphical editor and place it under the flowchart.



3. Select the point node in the graphical editor to open its **Properties** panel. Enter the run-time color expression for the shape in the **Color** property:
   `ATM.size() > 0 ? red : green`

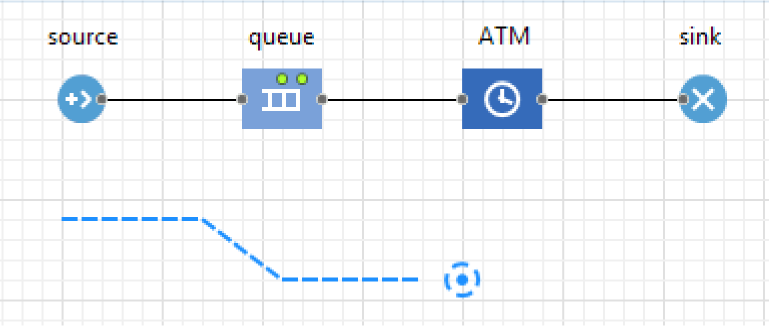
   Note that `ATM` here is the name of the **Delay** object we created. The expression determines the point node color at run time. The `size()` function returns the number of agents currently being processed. The color will be red, if a customer is served at this time, and green otherwise.
4. Click the *delay* block, called `ATM`, in the flowchart to open its **Properties** view.
5. Select the *point* node we have drawn in the **Agent location** option. You can either click the down arrow and select the point node from the list of appropriate objects, or you can click the button, located on the right, to select this space markup shape from the graphical editor (the rest of the elements in the editor will be greyed out).



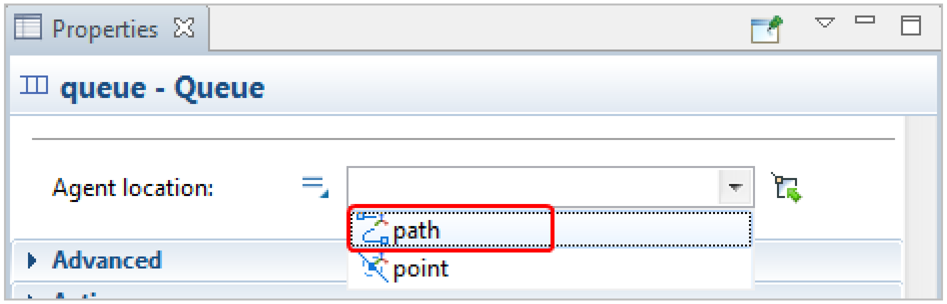
Agent location: point

Set up space markup for the queue

1. Draw the queue as a path. First, open the **Space Markup** palette in the **Palette** view.
2. Double-click the element **Path** to switch to the drawing mode.
3. Click in the graphical editor to put the first point of the path. Do more clicks to add turning points. Finish drawing with a double-click.



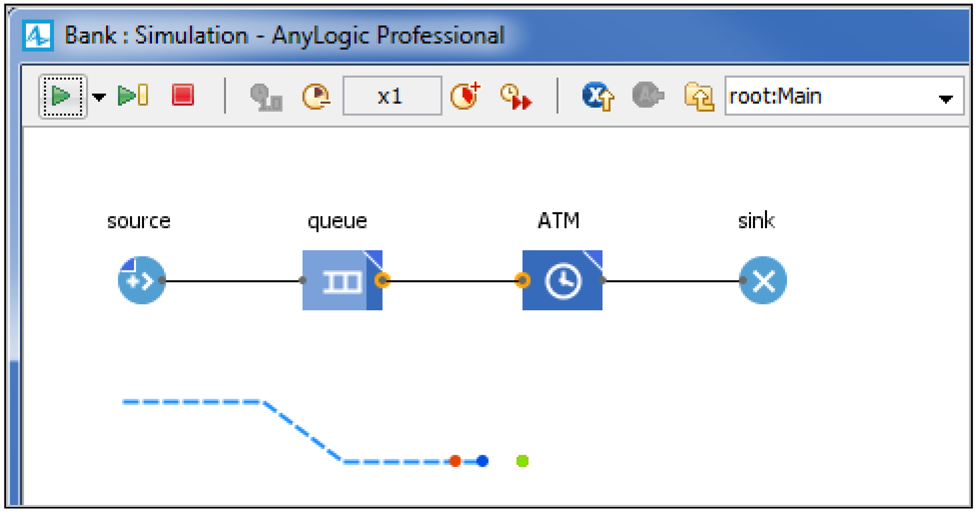
4. Click the *queue* block in the flowchart and go to its **Properties** view.
5. Select the *path* we have drawn in the **Agent location** option. You can either click the down arrow and select the path from the list of appropriate objects, or you can click the button, located on the right, to select this space markup shape from the graphical editor (the rest of the elements in the editor will be greyed out).



Agent location:
path
point

Now you can run the model and observe its behavior. If you want to speed up the simulation significantly, switch to virtual time mode by clicking the **Toggle real/virtual time mode** toolbar button. Switching to virtual time mode allows you to view simulation run at its maximum speed. Therefore, you can simulate a long period of time.

Note that when the ATM station is serving a customer, its animation shape becomes red, and when it is idle, it is green.



Adding 3D animation

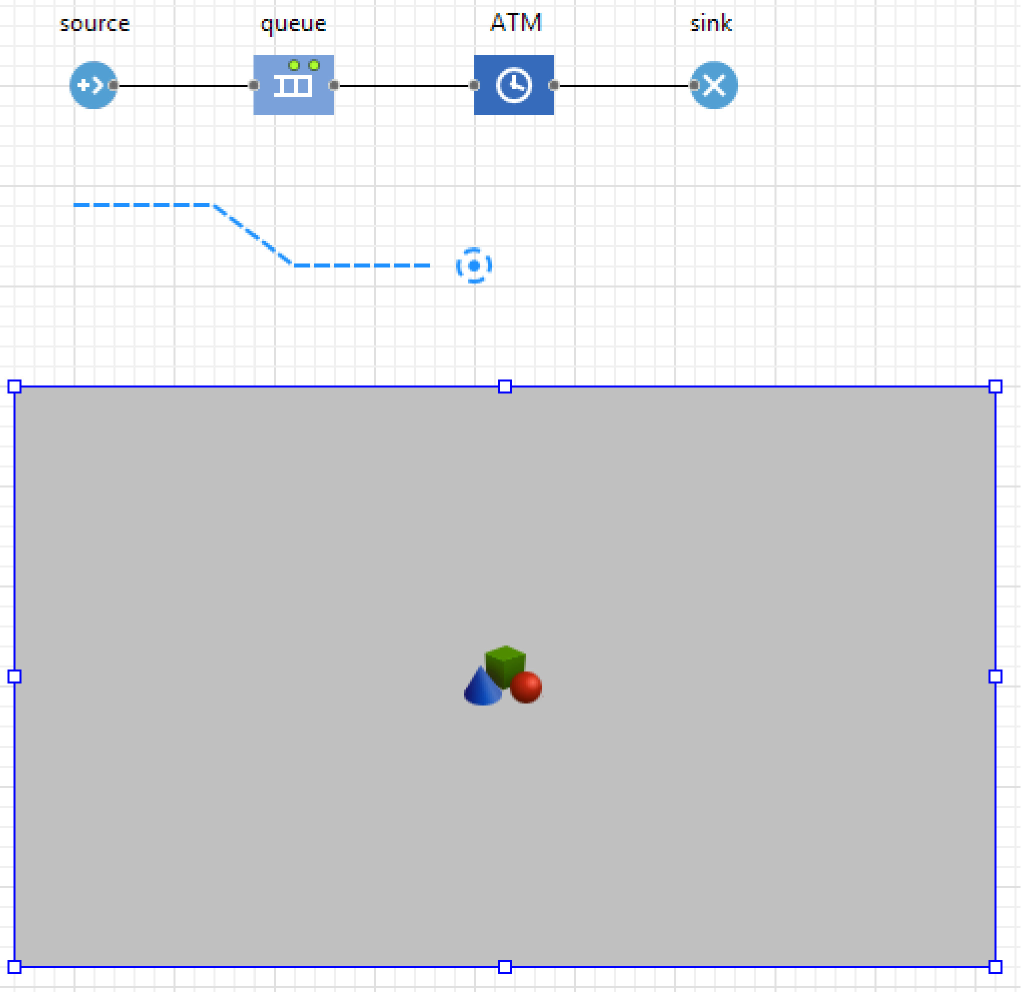
Both of the elements we have drawn by default are shown in 2D and 3D. You can check this property in the **Advanced** section of their properties view. Now we want to create 3D animation for our model.

First of all you should add 3D window on the diagram of your agent type.

> *3D window* plays the role of a placeholder for 3D animation. It defines the area on the presentation diagram where 3D animation will be shown at runtime.

Add 3D window

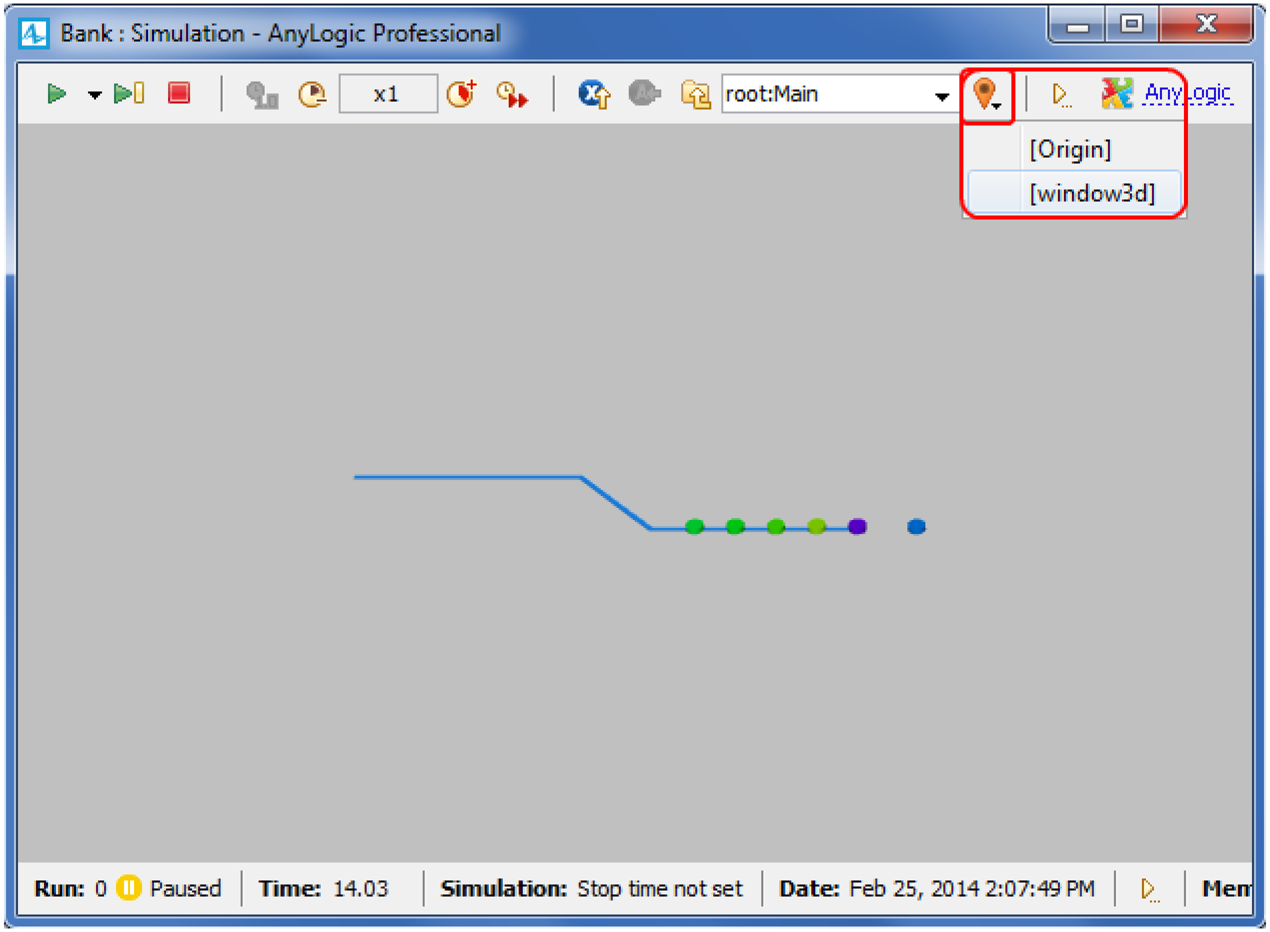
1. Drag the **3D Window** element from the **3D** section of the **Presentation** palette to the graphical editor.
2. The grey area will appear on the screen. Locate it where you want your 3D presentation to be shown at the model runtime:



Navigating through 3D animation

Now you can run your model and observe simple 3D animation.

1. Click the toolbar button **Navigate to view area...** and select **[window3D]**.



Run: 0 Paused | Time: 14.03 | Simulation: Stop time not set | Date: Feb 25, 2014 2:07:49 PM | Mem

2. Navigate through the 3D scene using the commands described below:

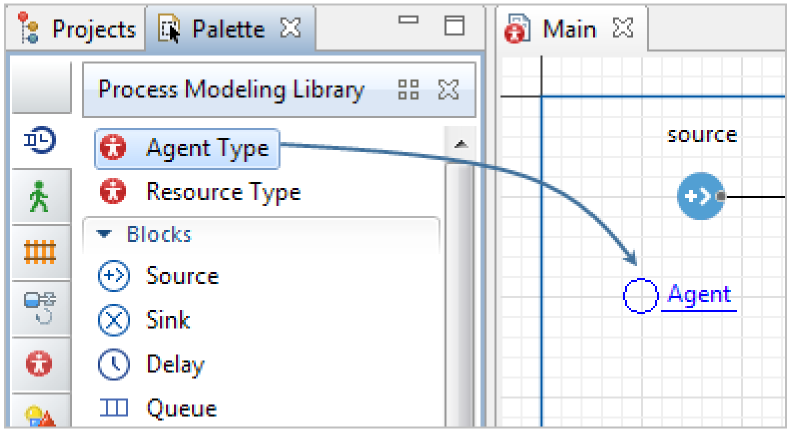
In order to	Use the mouse like described here
Move the scene	1. Press the left mouse button in the 3D view and hold the mouse button pressed. 2. Move the mouse in the required direction.
Rotate the scene	1. Press **`Alt`** key (Mac OS: `Option` key) and hold it pressed. 2. Click in the 3D scene window and, while holding `Alt` and the left mouse button down, 3. Move the mouse in the required rotation direction.
Zoom in/out the scene	1. Scroll the mouse wheel in the 3D window away from / towards you.

Adding 3D objects

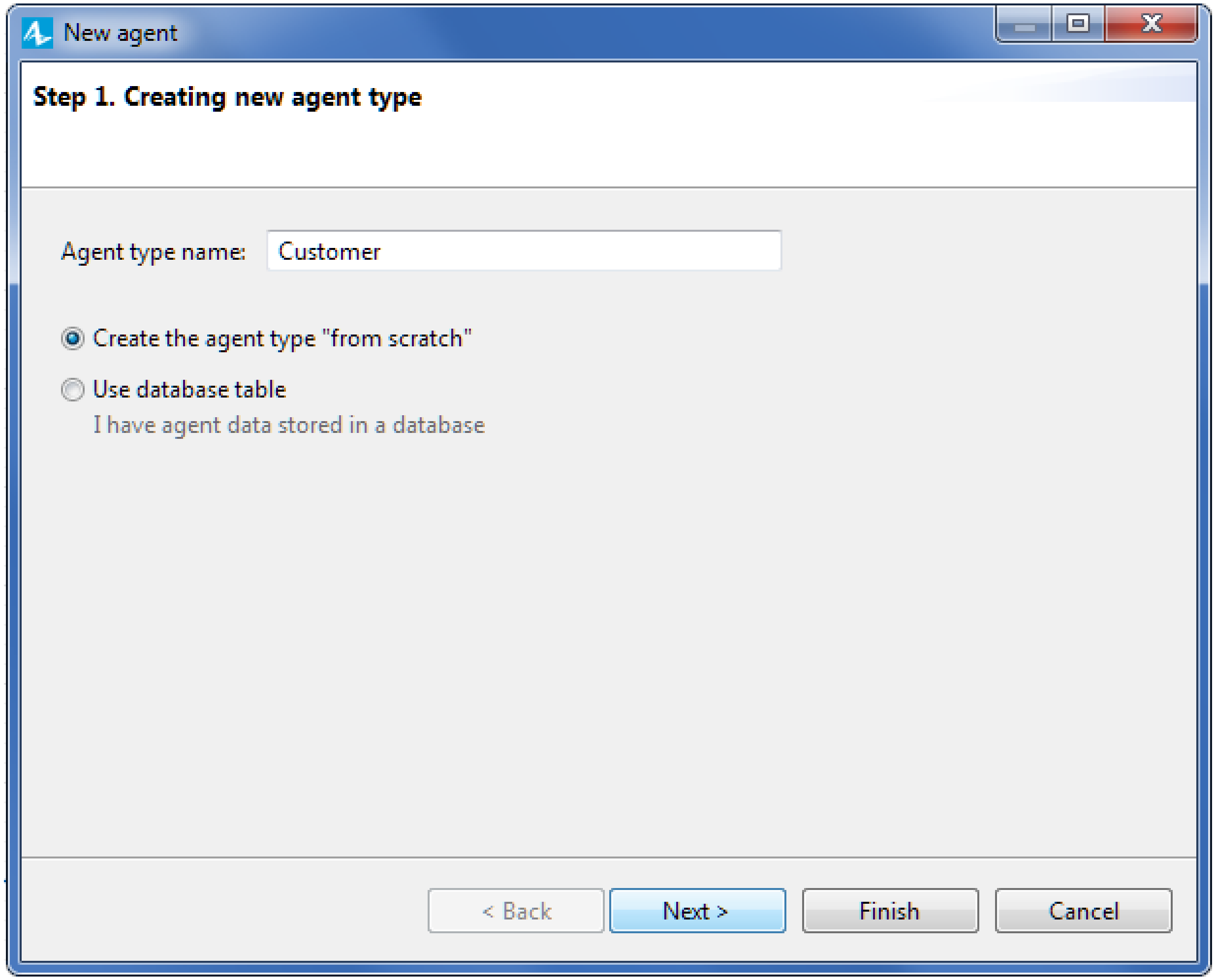
Now we want to add a customer figure. By default the customers are drawn as coloured dots and shown as colored cylinders in 3D animation. We want to create our custom type of client and animate it in 3D. We will create a new agent type for this purpose.

Create a new agent type

1. Open the **Process Modeling Library** in the **Palette** view.
2. Drag the element **Agent Type** into the graphical editor.



3. The **New agent** wizard will open on the **Creating new agent type** step. Enter *Customer* as the **Agent type name**, and leave the **Create the agent type "from scratch"** selected. Press **Next**.



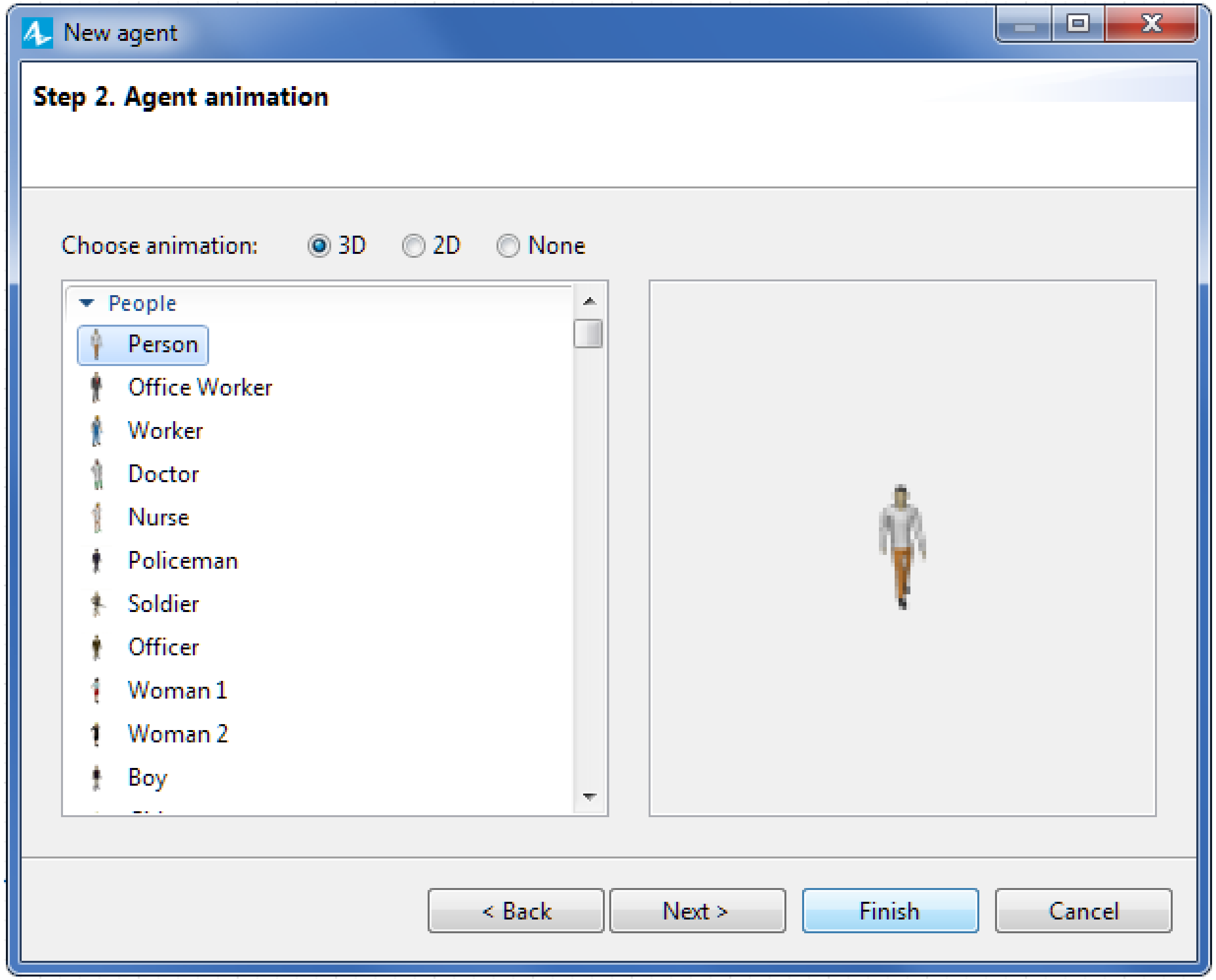
Step 1. Creating new agent type

Agent type name: Customer

● Create the agent type "from scratch"

○ Use database table
I have agent data stored in a database

4. In the next step select **3D** as the animation type and select *Person* from the list of the 3D figures.



New agent

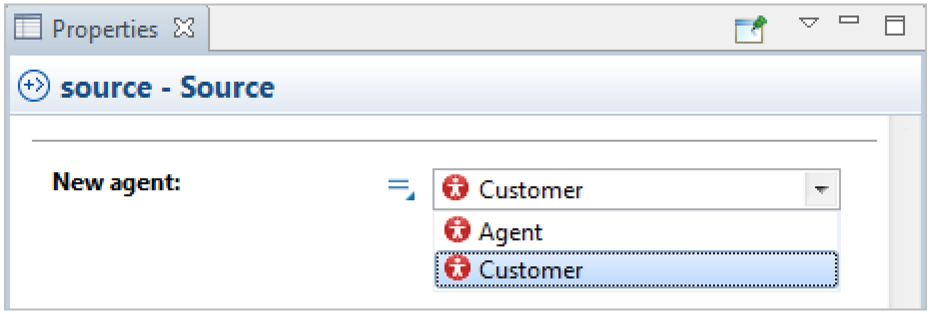
Step 2. Agent animation

Choose animation: ● 3D ○ 2D ○ None

5. Click **Finish**. The new `Customer` diagram will open. You can find the *Person* 3D figure in the axis origin. Switch back to the `Main` diagram.

Configure flowchart to use the new type

1. On the `Main` diagram, select the block *source* in the graphical editor.
2. Choose `Customer` in the **New agent** drop-down list.

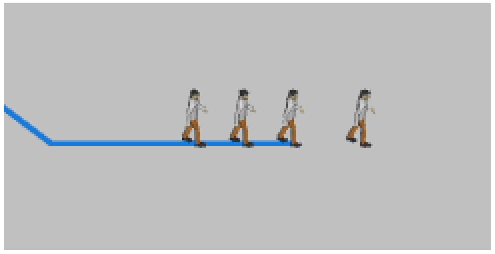


Properties

source - Source

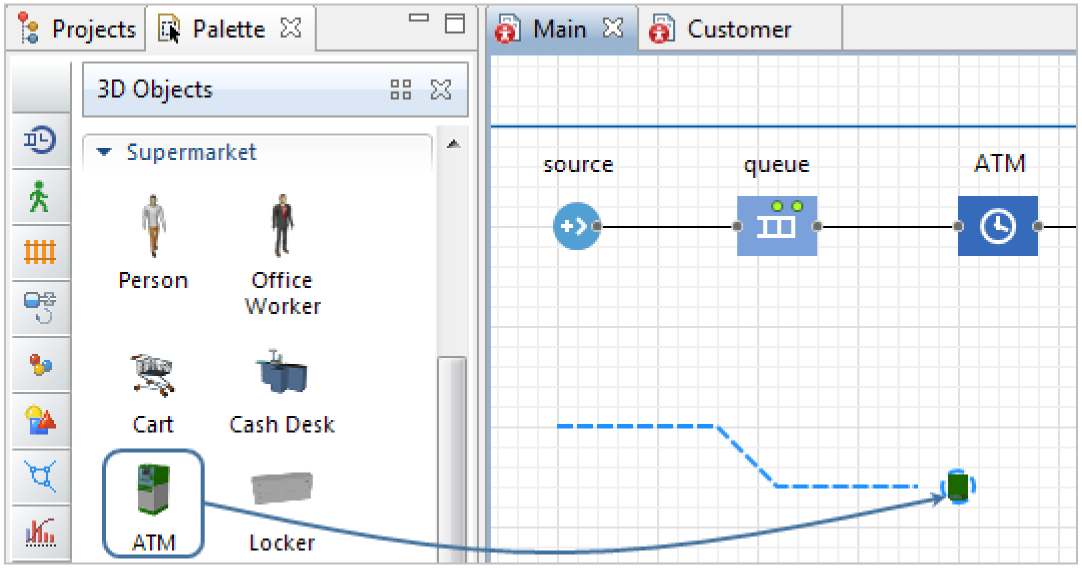
New agent: Customer

3. Run the model and switch to 3D view to see our customers moving in the queue.



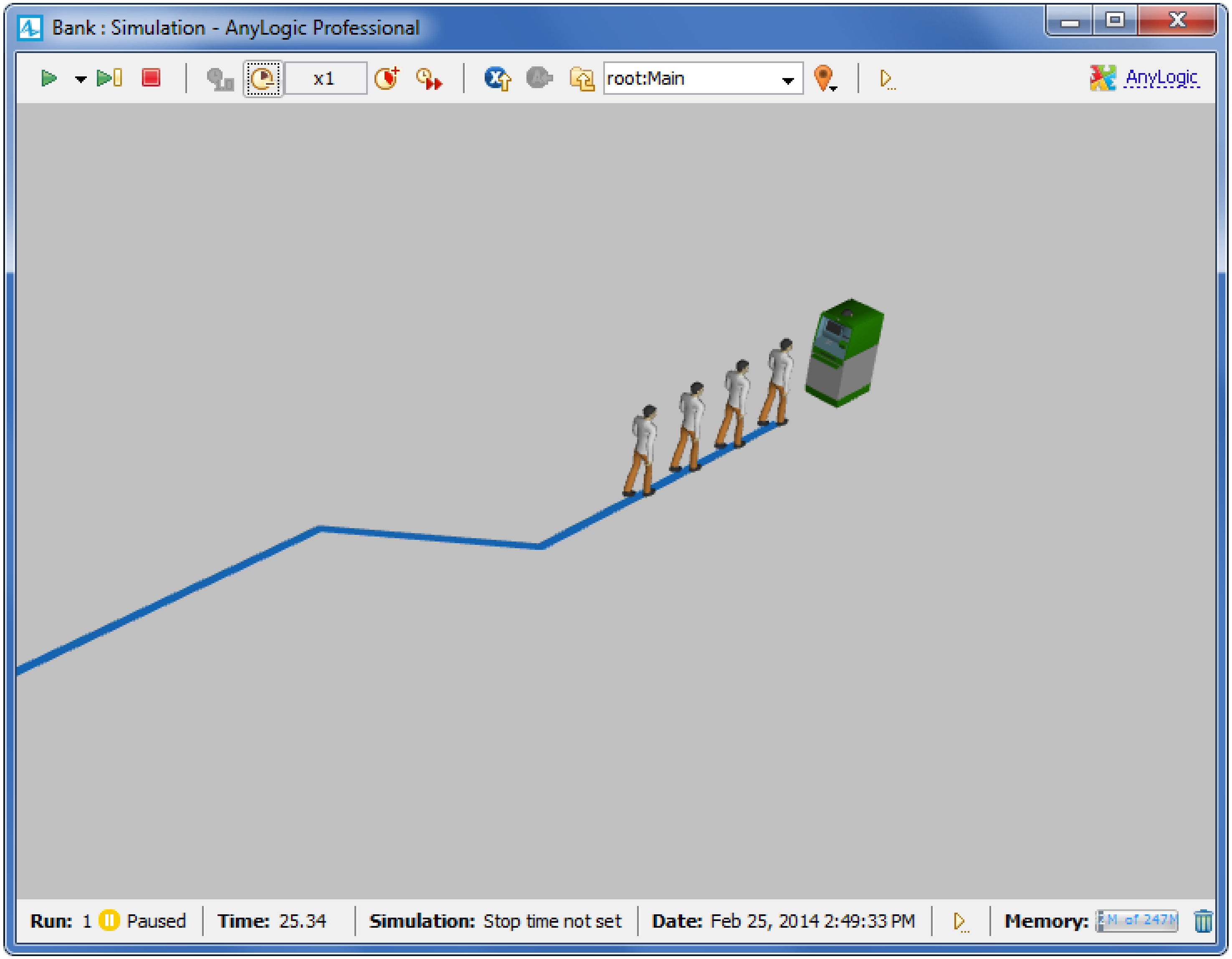
Add an ATM figure

1. Open the **3D Objects** palette in the **Palette** view.
2. Drag the **ATM** 3D figure from the **Supermarket** section of this palette onto the point node shape in the graphical editor.



3. If you run the model now and check 3D animation in **window3D** mode, you will notice that our ATM does not face the customers' flow and we need to rotate it.

4. Select the *atm* 3D object in the graphical editor and open the section **Position** in its properties view.
5. Choose *0* degrees from the drop-down list of the **Rotation Z** option.
6. Run the model to double-check that the ATM is facing the customers now.



Run: 1 Paused | **Time:** 25.34 | **Simulation:** Stop time not set | **Date:** Feb 25, 2014 2:49:33 PM | **Memory:** M of 247M

Reference model: Bank - Phase 2

Phase 1, Creating a simple model

Phase 3, Adding tellers

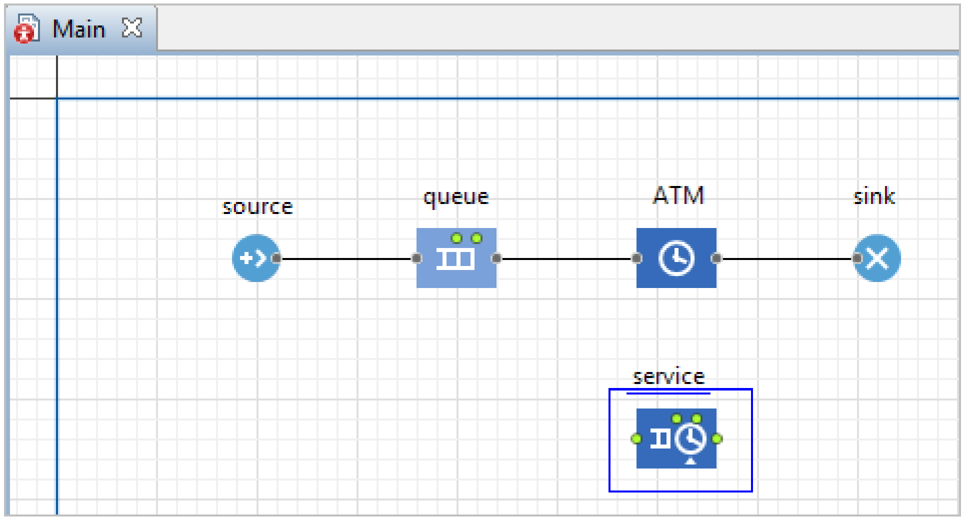
Phase 3. Adding Tellers

Now we will create another part of the system by adding tellers that are working at the bank. Now some clients will come to see tellers, some – to access the ATM. We can model tellers using delays in the same way as we modeled ATM. However, modeling tellers using resources is much more convenient. *Resource* is a special unit that can be possessed by an agent. Only one agent can possess a resource at a time; therefore agents compete for resources.

Modifying the flowchart

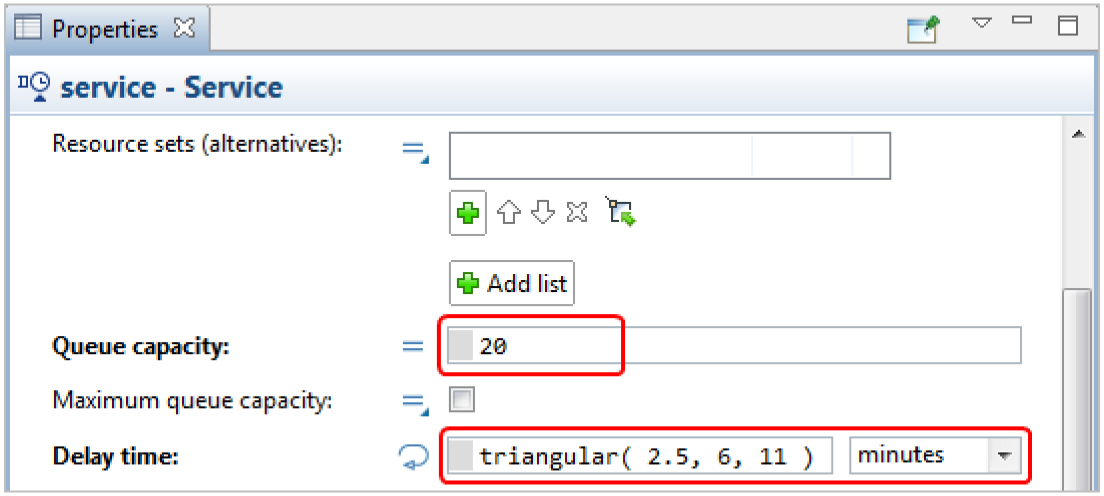
Simulate service

1. Open the **Process Modeling Library** in the **Palette** view and drag the **Service** block onto our `Main` diagram.



Service seizes resource units for the agent, delays the agent, and releases the seized units.

2. Go to the **Properties** view of the *service* block.
3. Modify the object properties:
   - There is one queue for all tellers. Set up **Queue capacity** to be of 20 places.
   - We assume that service time is triangularly distributed with the min value of 2.5, average value of 6, and the max value of 11 minutes. Set **Delay time**: `triangular( 2.5, 6, 11 )`



service - Service

Resource sets (alternatives):

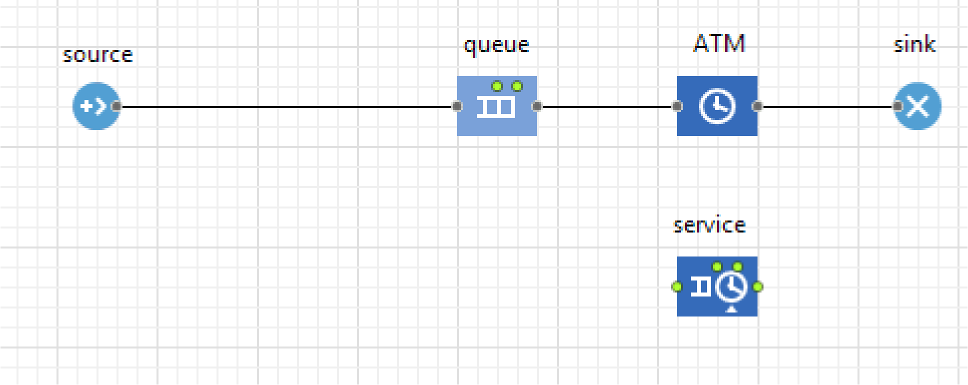
Queue capacity: 20

Maximum queue capacity: □

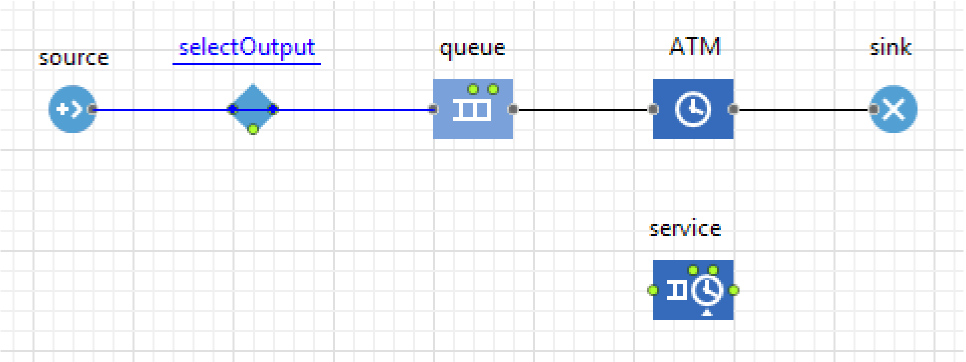
Delay time: triangular(2.5, 6, 11) minutes

Simulate decision making

1. Move the objects *queue*, *ATM* and *sink* to the right to make space for one new object between *source* and *queue*.

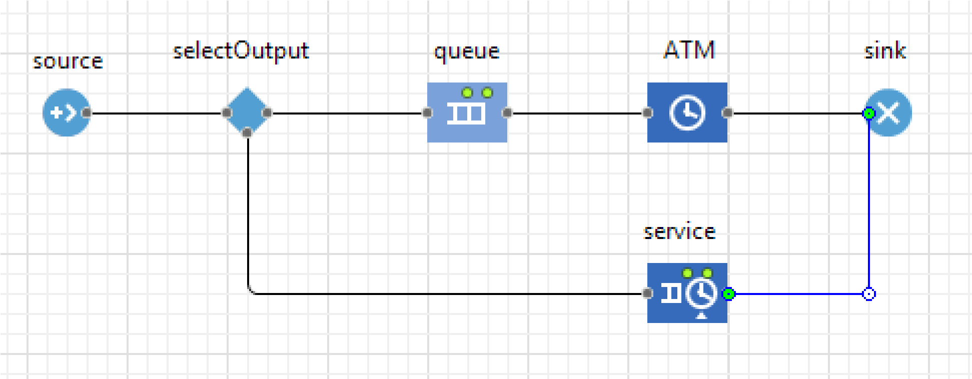


2. Open the **Process Modeling Library** in the **Palette** view and add the [SelectOutput]() block in the resulting space. When you place the object on the connector, it will automatically get built in.



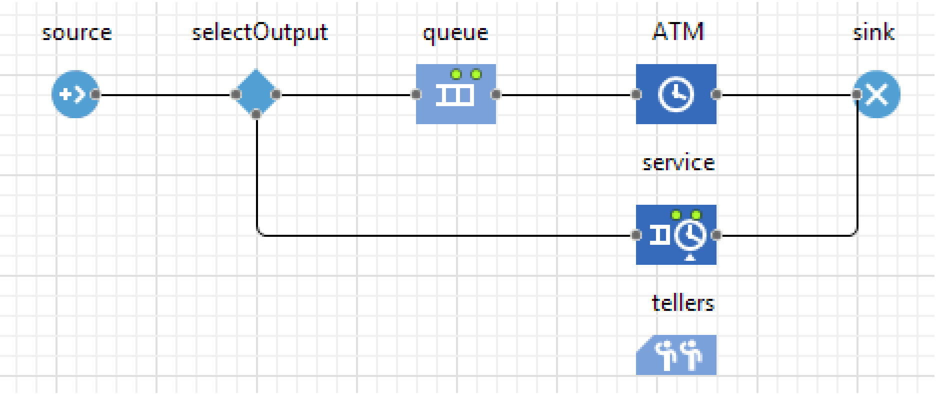
SelectOutput object is a decision making block. The agent arrived at the object is forwarded along one of two output ports depending on the user-defined condition.

3. Select *selectOutput* in the flowchart and go to its **Properties** view. Choose the option *If condition is true* for the **Select True Output** parameter. Make sure that **Condition** is `randomTrue( 0.5 )`.
   This agent routing condition defines that the number of customers competing for ATM and teller service will be approximately equal.
4. Connect *selectOutput* and *service* with other blocks as shown in the figure:

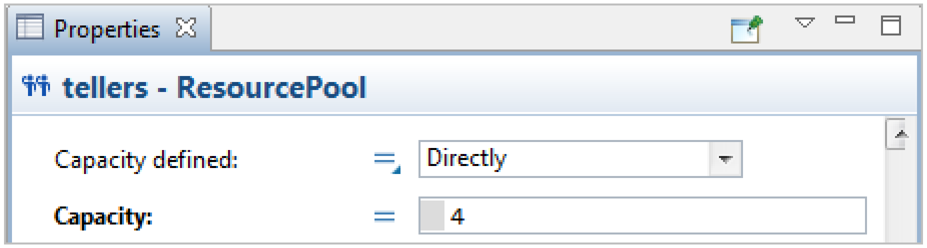


Add resources for the service

1. Open the **Process Modeling Library** in the **Palette** view and drag the **ResourcePool** block onto our `Main` diagram. **ResourcePool** object is storage for resource units.
2. Place it under *service* and go to its **Properties** view.
3. Name the object *tellers*.



4. Specify that this resource object has only four resource units, that means, define its **Capacity**: 4.



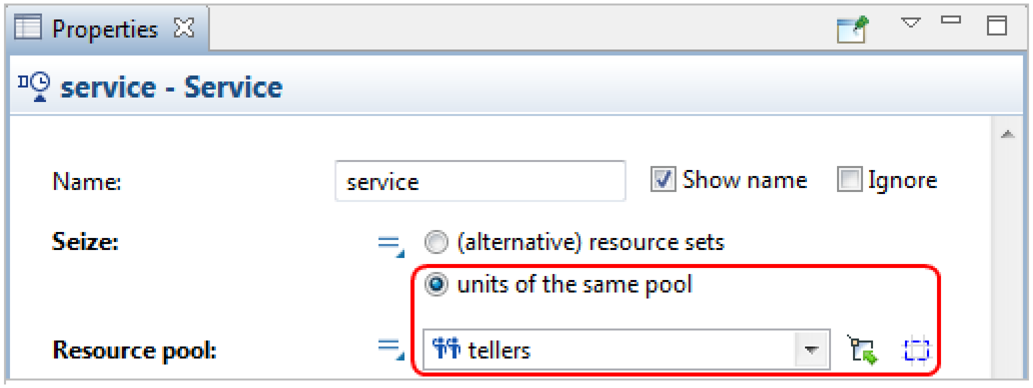
Properties

tellers - ResourcePool

Capacity defined: Directly

Capacity: 4

5. **ResourcePool** object should be connected to resource seizing and releasing objects (**Service** in our case). So we need to modify the properties of the *service* object.
6. Select *service* in the flowchart to open its properties. Choose the option *units of the same pool* the parameter **Seize**. Then specify the resource pool we have created in the option **Resource Pool**. You can either click the down arrow to select the resource pool object from the drop-down list, or you can click the button, located on the right, to select the object in the graphical editor (all inappropriate objects will be greyed out).



Properties

service - Service

Name: service ☑ Show name ☐ Ignore

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

Resource pool: tellers

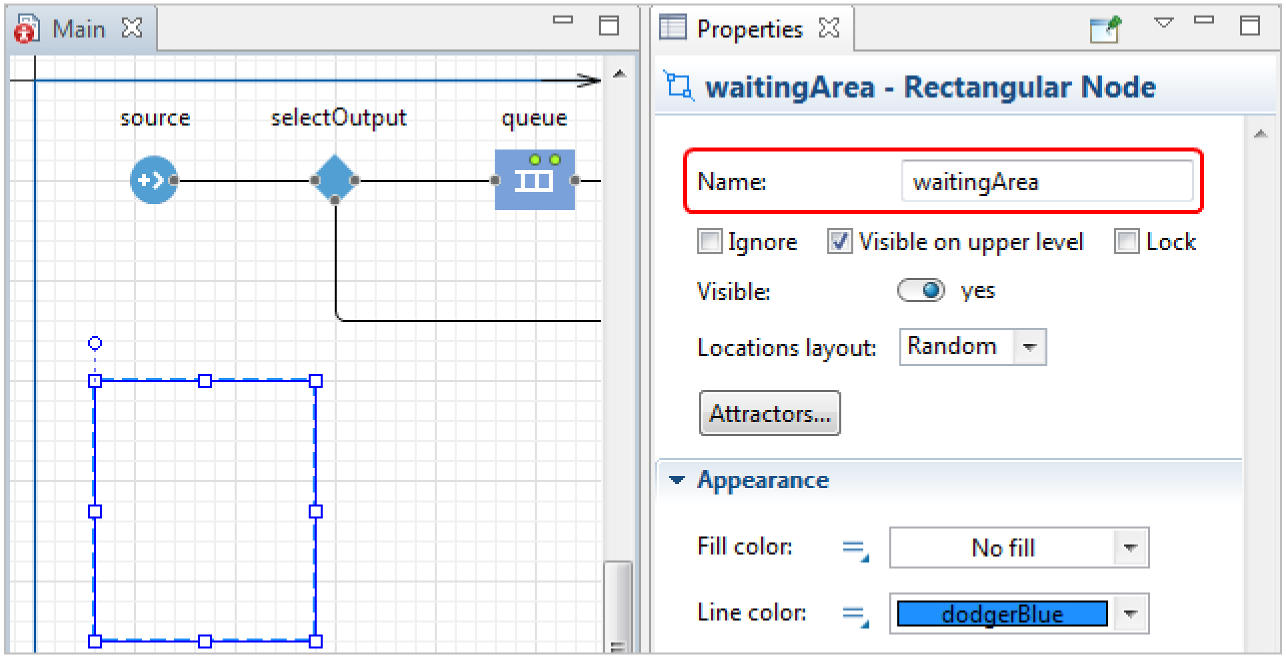
7. Now since the model has changed, we need to alter the model animation as well.

Adding space markup shapes

Now we want to draw the area for queueing and a place to get serviced for our clients.

Set up space markup for the queue to tellers

1. This time we will draw a waiting area using a rectangular node. First, open the **Space Markup** palette in the **Palette** view.
2. Double-click the element **Rectangular node** to switch to *the drawing mode*.
3. Click in the graphical editor and drag the rectangle without releasing the mouse button. Release when you have a rectangular node of the required form. You can edit its form later as you need.
4. Name the node *waitingArea*.



waitingArea - Rectangular Node

Name: waitingArea

☐ Ignore ☑ Visible on upper level ☐ Lock

Visible: yes

Locations layout: Random

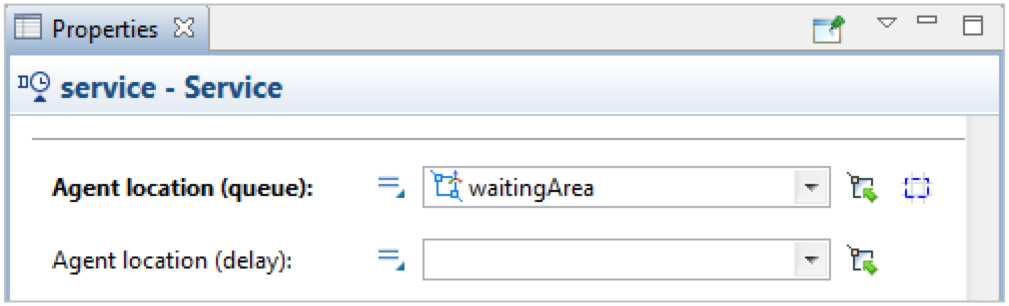
Attractors...

Appearance

Fill color: No fill

Line color: dodgerBlue

5. Click the *service* block in the flowchart and go to its **Properties** view.
6. Select the node *waitingArea* we have drawn in the **Agent location (queue)** option.



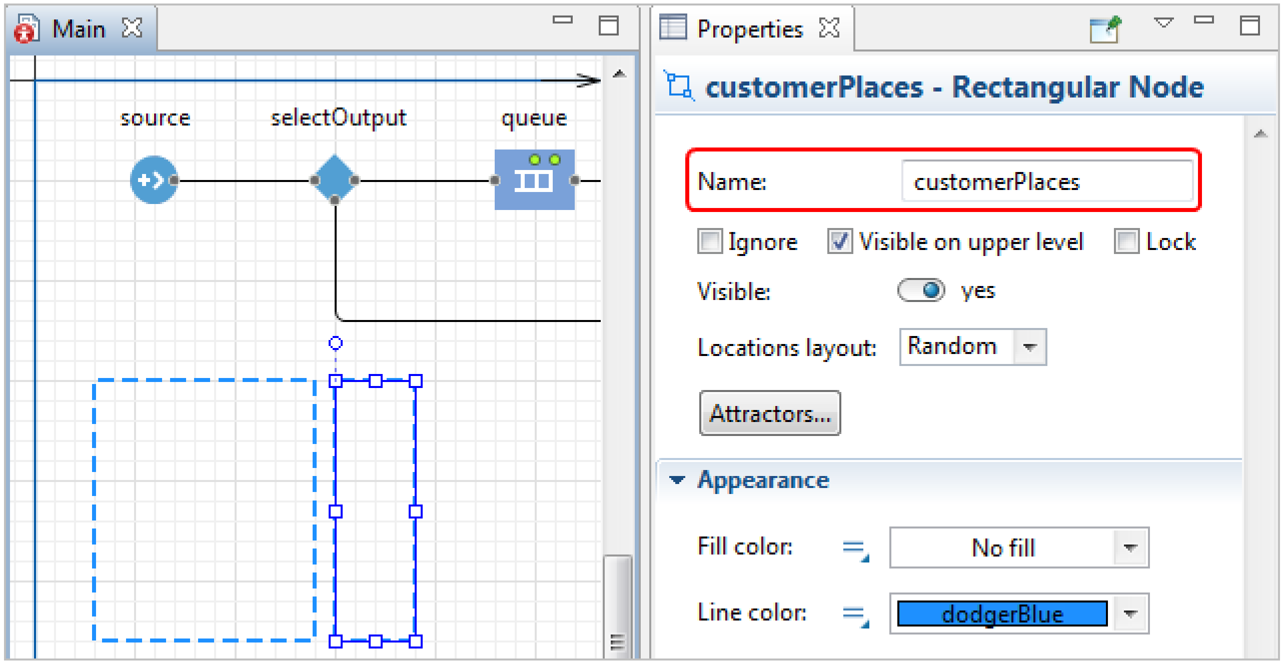
service - Service

Agent location (queue): waitingArea

Agent location (delay):

Set up space markup for the customers

1. The customers need a place to stand somewhere while they are getting serviced by tellers. We will draw an area for this purpose using a rectangular node.
2. First, open the **Space Markup** palette in the **Palette** view, then double-click the element **Rectangular node** to switch to *the drawing mode*.
3. Click in the graphical editor and drag the rectangle without releasing the mouse button. Release when you have a rectangular node of the required form. You can edit its form later as you need.
4. Name the node *customerPlaces*.



customerPlaces - Rectangular Node

Name: customerPlaces

□ Ignore ☑ Visible on upper level □ Lock

Visible: yes

Locations layout: Random

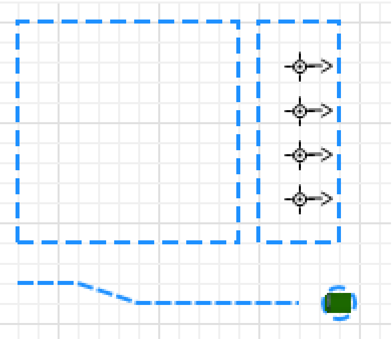
Attractors...

Appearance

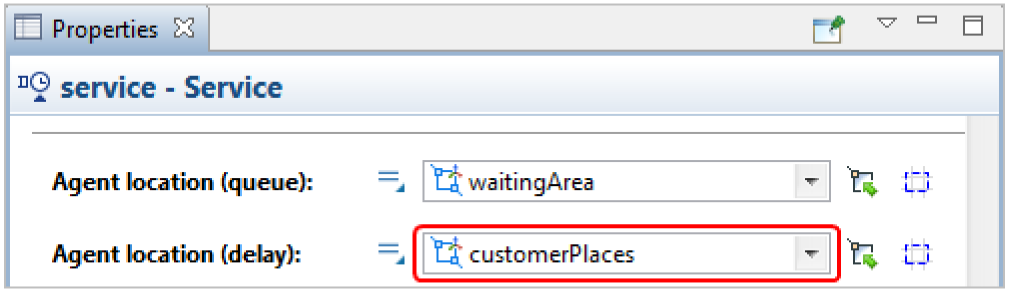
Fill color: No fill

Line color: dodgerBlue

5. We will use attractors to define the customers that are getting service. Select the node *customerPlaces* and click the button **Attractors...** in its properties. In the **Attractors** window that will pop-up, specify *4* for the creation mode **Number of attractors** and click **OK**. You will see that attractors appeared in the *customerPlaces* node with the even offset.



6. Now we need to refer to this area in the flowchart. Click the *service* object in the flowchart and go to its **Properties** view.
7. Select the node *customerPlaces* have drawn in the **Agent location (delay)** option.



Properties

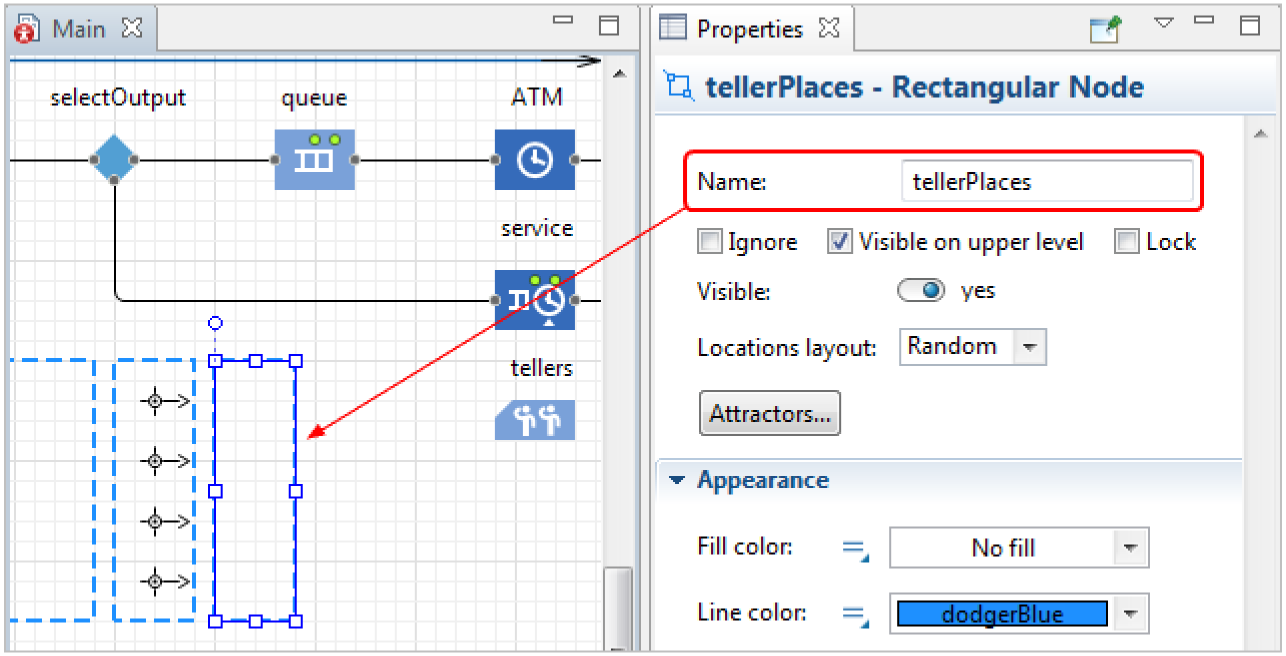
service - Service

Agent location (queue): waitingArea

Agent location (delay): customerPlaces

Set up space markup for the tellers

1. In the previous step, we used a point node to draw the ATM. Since we have 4 tellers this time, we will use a rectangular node to draw this service area.
2. First, open the **Space Markup** palette in the **Palette** view, then double-click the element **Rectangular node** to switch to *the drawing mode*.
3. Click in the graphical editor and drag the rectangle without releasing the mouse button. Release when you have a rectangular node of the required form. You can edit its form later as you need.
4. Name the node *tellerPlaces*.



Properties

tellerPlaces - Rectangular Node

Name: tellerPlaces

□ Ignore ☑ Visible on upper level □ Lock

Visible: yes

Locations layout: Random

Attractors...

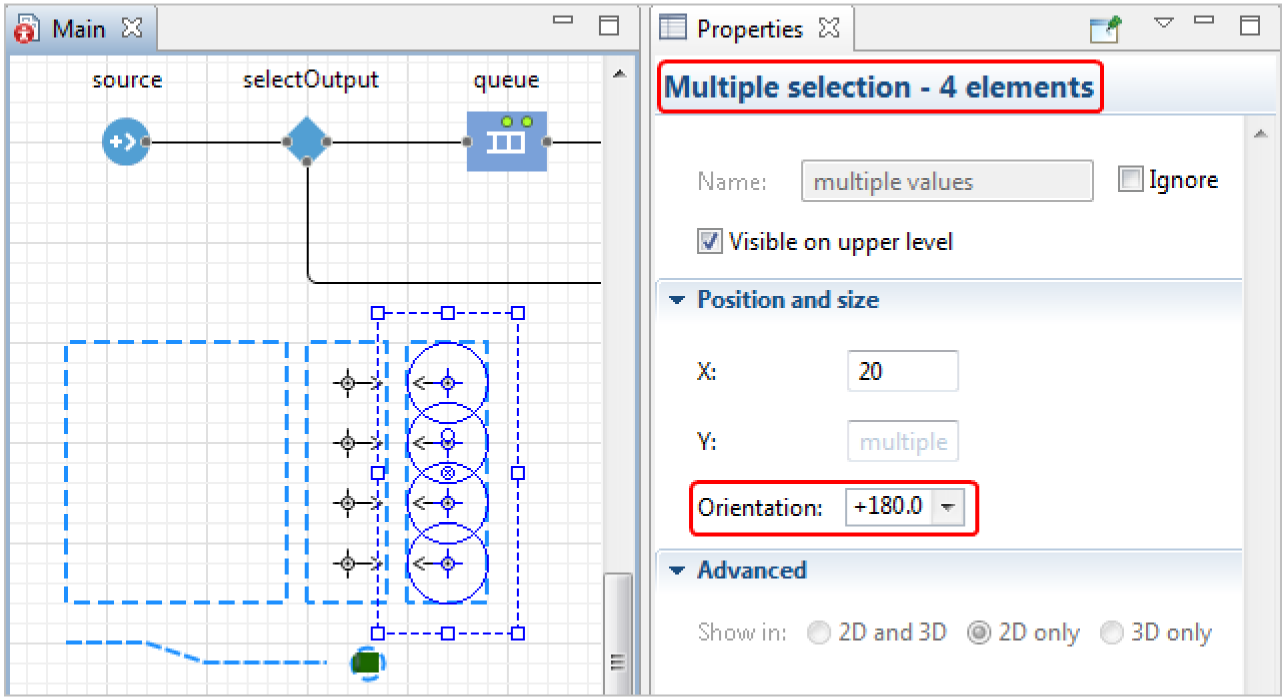
Appearance

Fill color: No fill

Line color: dodgerBlue

5. We will use attractors to define the tellers. Select *tellerPlaces* and click the button **Attractors...** in its properties. In the **Attractors** window that will pop-up, specify *4* for the creation mode **Number of attractors** and click **OK**.

6. You will see that attractors appeared in the *tellerPlaces* node with the even offset, but they are facing wrong direction. Select all attractors by Shift+clicking and go to their properties. In the section **Position and size**, change the **Orientation** parameter to *+180.0*.



Multiple selection - 4 elements

Name: multiple values □ Ignore

☑ Visible on upper level

Position and size

X: 20

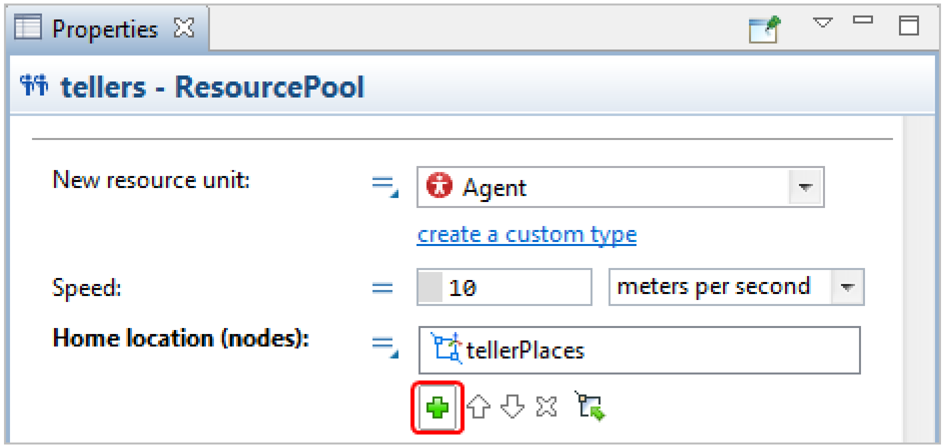
Y: multiple

Orientation: +180.0

Advanced

Show in: ○ 2D and 3D ● 2D only ○ 3D only

7. Click the *tellers* object in the flowchart and go to its **Properties** view.
8. Select the node *tellerPlaces* we have drawn in the **Home location (nodes)** parameter.



tellers - ResourcePool

New resource unit: Agent

create a custom type

Speed: 10 meters per second

Home location (nodes): tellerPlaces

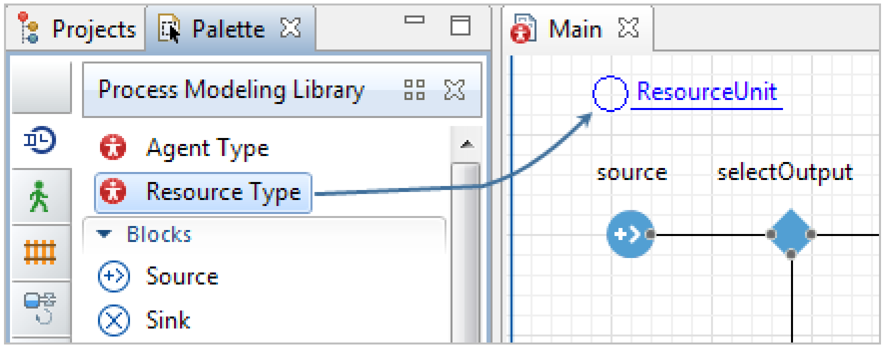
You can run the model now and observe how some customers are getting serviced at the ATM and some go to see tellers.

Adding 3D objects

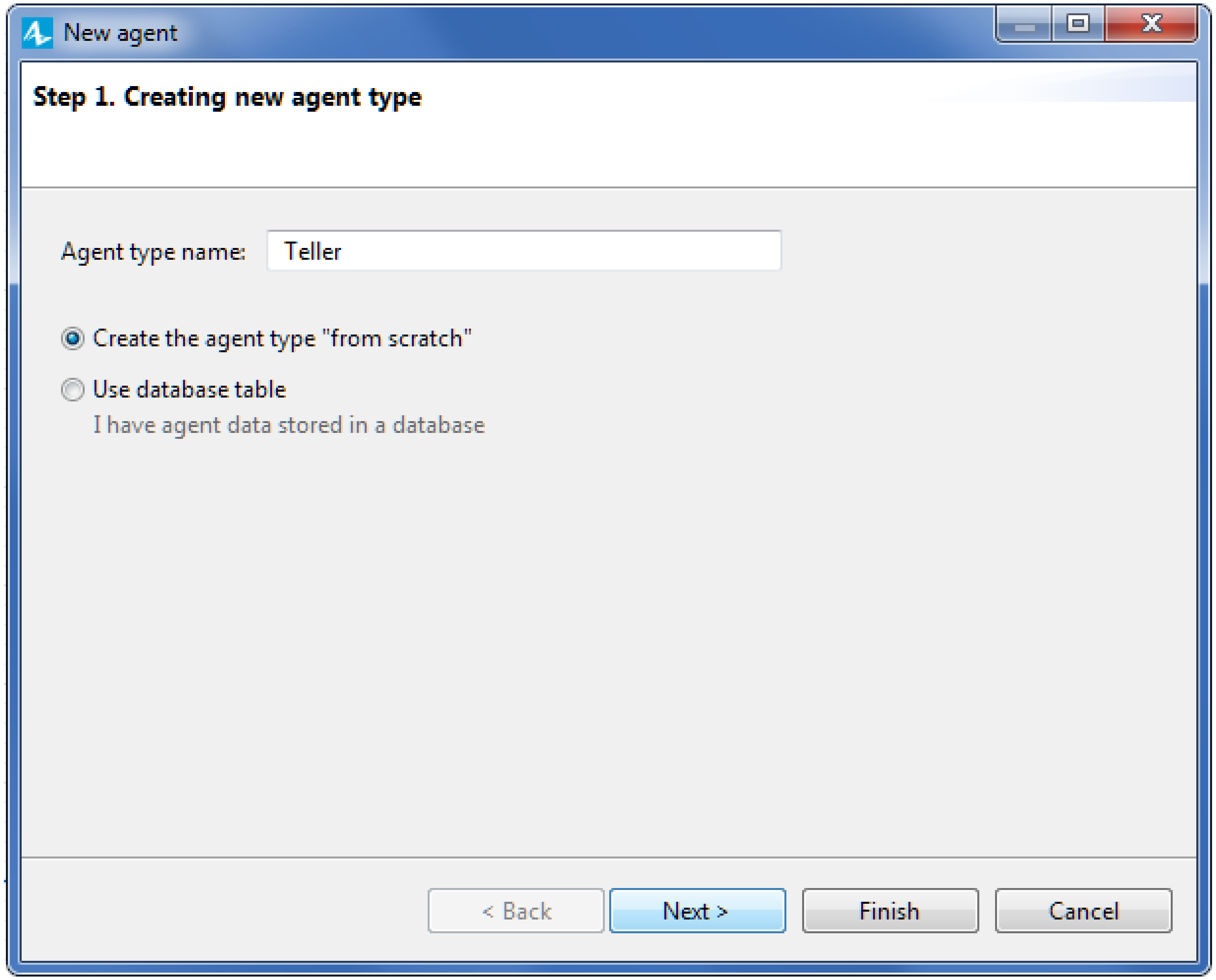
It is time to add teller 3D objects to our model. We will create a new resource type to animate tellers.

Create a new resource type

1. Open the **Process Modeling Library** in the Palette view.
2. Drag the element **Resource type** into the graphical editor.



3. The **New agent** wizard will open on the **Creating new agent** step. Enter *Teller* as the **Agent type name** and leave the **Create the agent type "from scratch"** selected. Press **Next**.



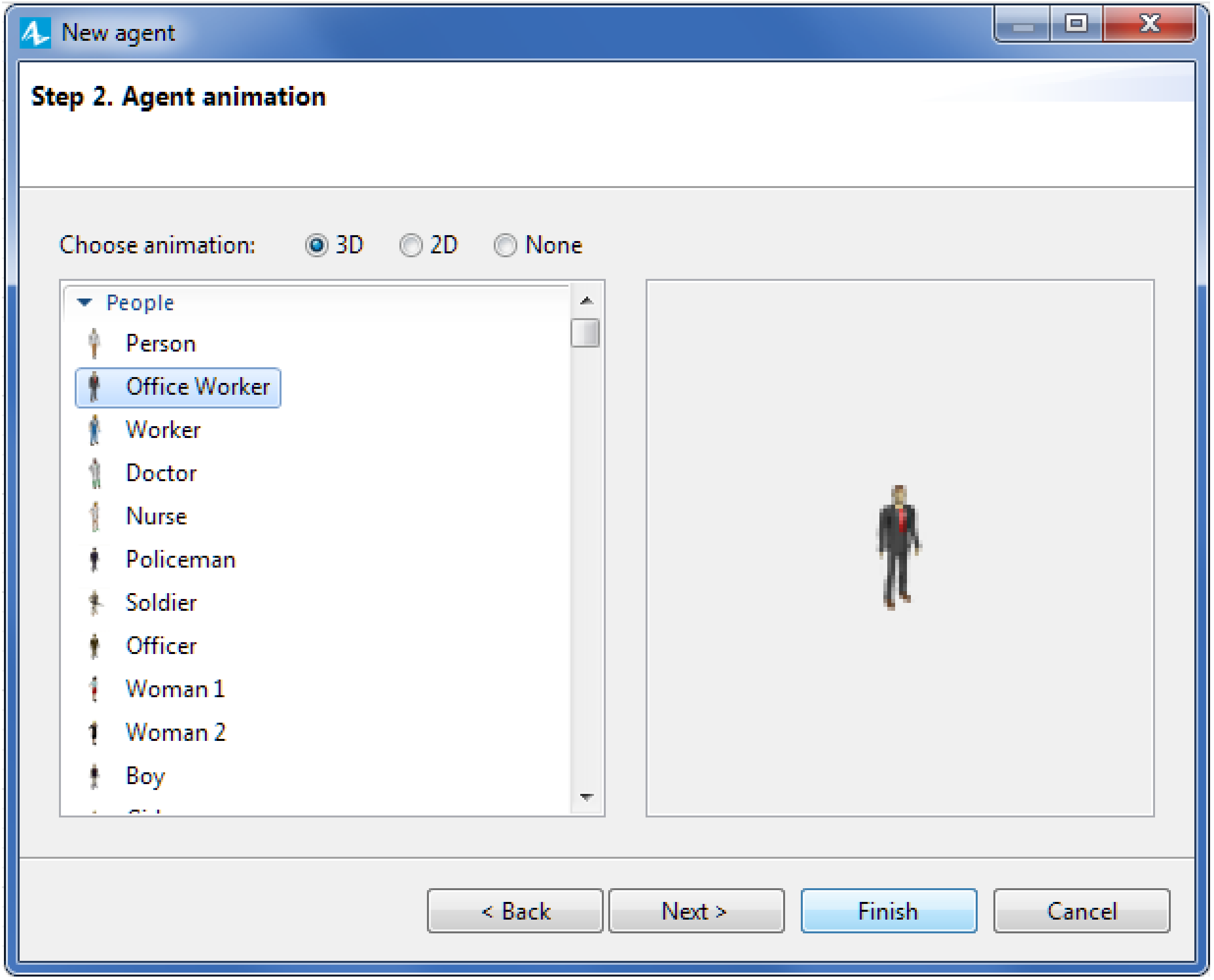
Step 1. Creating new agent type

Agent type name: Teller

● Create the agent type "from scratch"

○ Use database table
I have agent data stored in a database

4. In the next step select **3D** as the animation type and select *Office worker* from the list of the 3D figures.



New agent

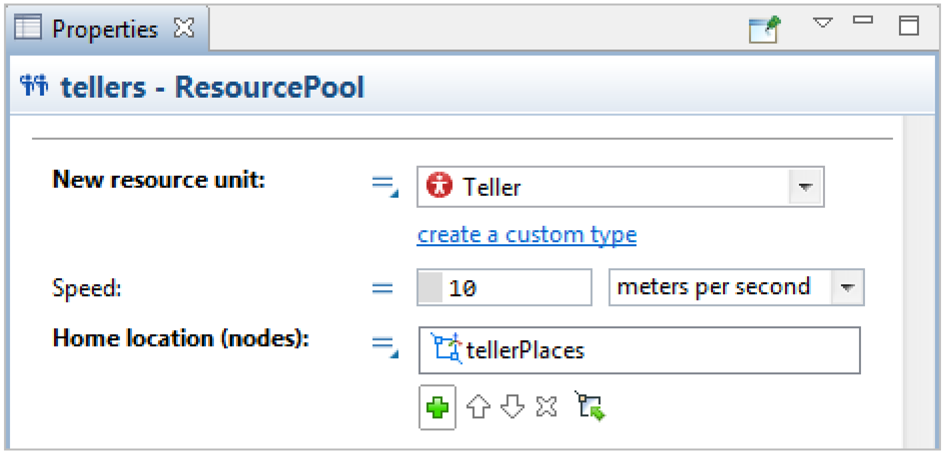
Step 2. Agent animation

Choose animation: ● 3D ○ 2D ○ None

5. Click **Finish**. The new `Teller` diagram will open. You can find the *Office worker* 3D figure in the axis origin. Switch back to `Main` diagram.

Configure flowchart to use the new resource type

1. On the `Main` diagram, select the block *tellers* in the graphical editor.
2. In its properties, specify `Teller` as the **New resource unit**.



Properties

tellers - ResourcePool

New resource unit: Teller

create a custom type

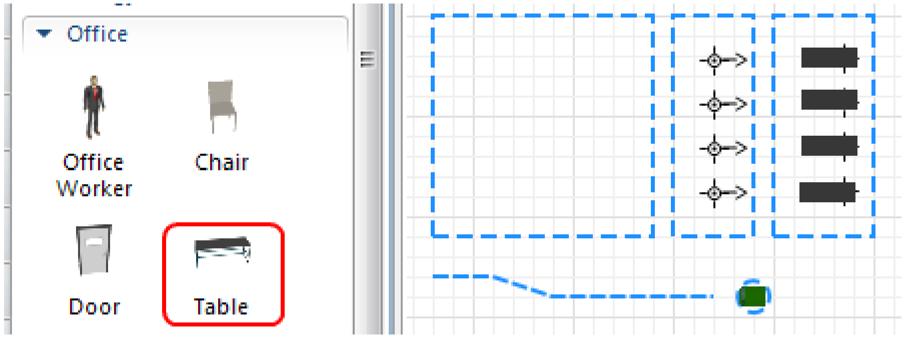
Speed: 10 meters per second

Home location (nodes): tellerPlaces

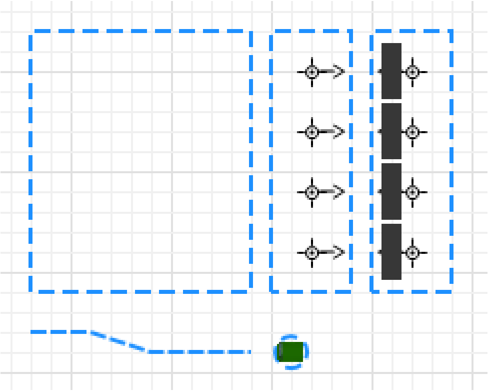
3. Run the model and observe customers and tellers.

Add tables for the tellers

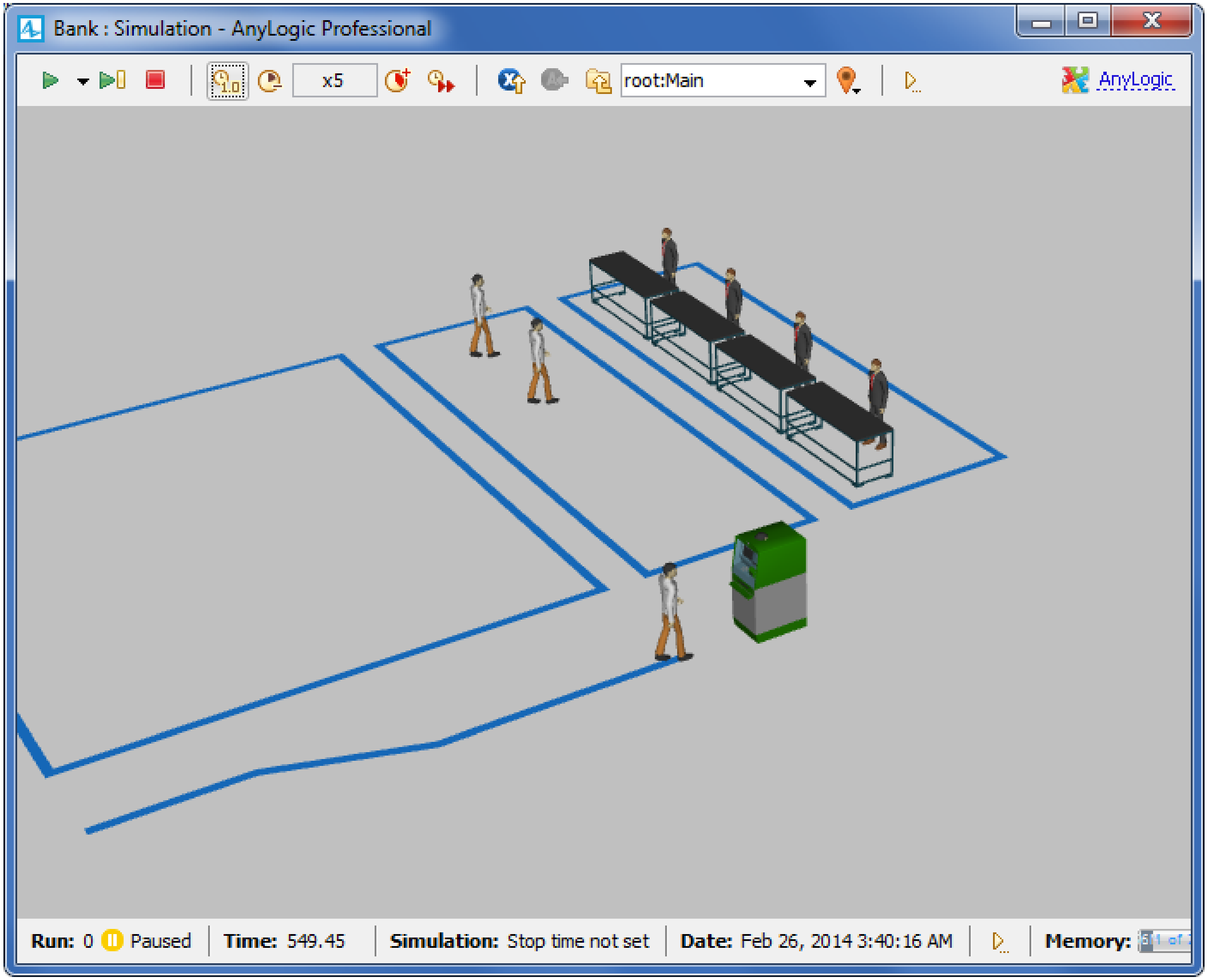
1. Open the **3D Objects** palette in the **Palette** view.
2. Drag four **Table** 3D figures from the **Office** section of this palette onto the node shape called *tellerPlaces* in the graphical editor.
3. Place them at the attractors since attractors are the places where the tellers stand.



4. You can see that their orientation is wrong. Select all tables by Shift-clicking and go to the **Properties** view.
5. In the section **Position**, change the parameter **Rotation** to *-90.0* degrees.
6. If necessary, rearrange all eight attractors and four tables so that they are reasonably lined up.



Now you can run the model and observe in 3D how some customers go to the ATM and other get service at the tellers tables.



Run: 0 Paused | **Time:** 549.45 | **Simulation:** Stop time not set | **Date:** Feb 26, 2014 3:40:16 AM | **Memory:**

Reference model: Bank - Phase 3

Phase 2, Creating model animation

Phase 4, Adding statistics collection

Tutorials > Bank Office (Queueing system)

Phase 4. Adding Statistics Collection

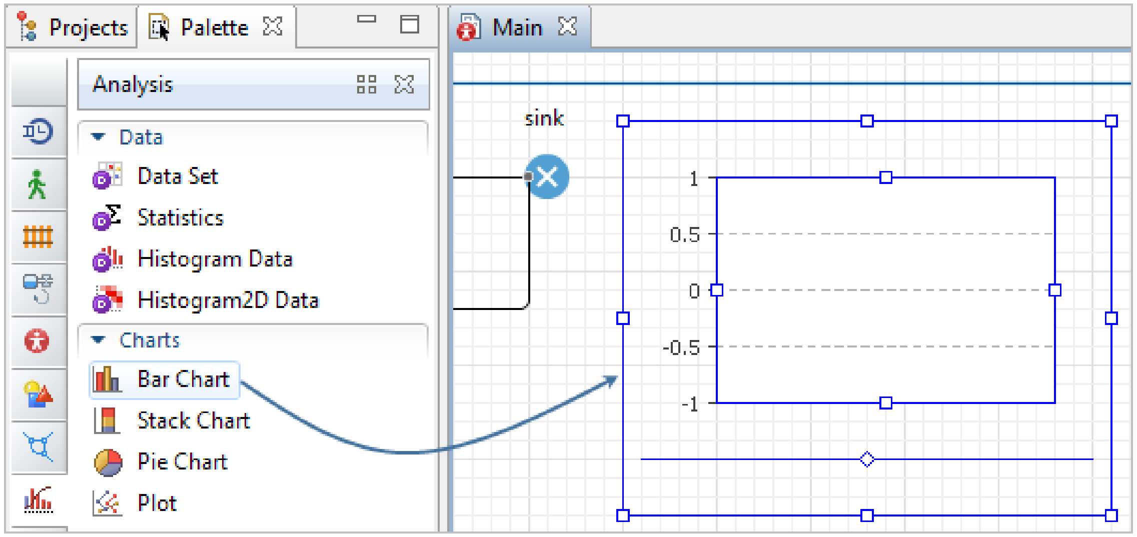
With AnyLogic, you can collect complicated statistics whenever you need them. The objects of the Process Modeling Library are already capable of collecting the basic statistics. All you need is to turn the statistics collection for the object on, as it is disabled by default to speed up the model execution. We can view the statistics collected for the flowchart objects with charts.

We want to observe how mean ATM utilization and mean queue length change with time.

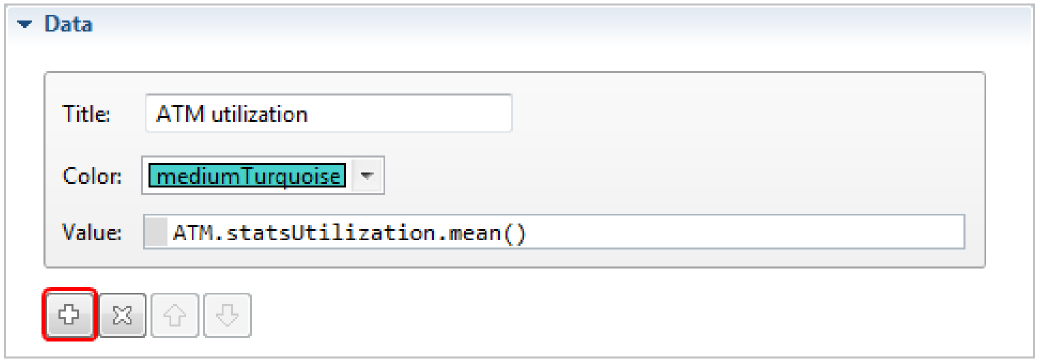
Collecting utilization statistics

Add a bar chart to indicate mean ATM utilization

1. Open the **Analysis** palette of the **Palette** view. This palette contains charts and data objects used for collecting data and performing various statistical analysis on them. Drag the **Bar Chart** element from the stencil into the graphical editor.



2. Go to the **Data** section in **Properties** of the chart. Click **Add data item** to add data item to be displayed by this chart.
3. Modify the data item's **Title**: *ATM utilization*.
4. Type `ATM.statsUtilization.mean()` as the **Value** of the data item. `ATM` is the name of the **Delay** object we created. Each **Delay** object has `statsUtilization` data set that collects statistics on the object utilization. The `mean()` is the function that returns the mean value measured. You can use other functions to get statistical values, such as `min()` and `max()`.



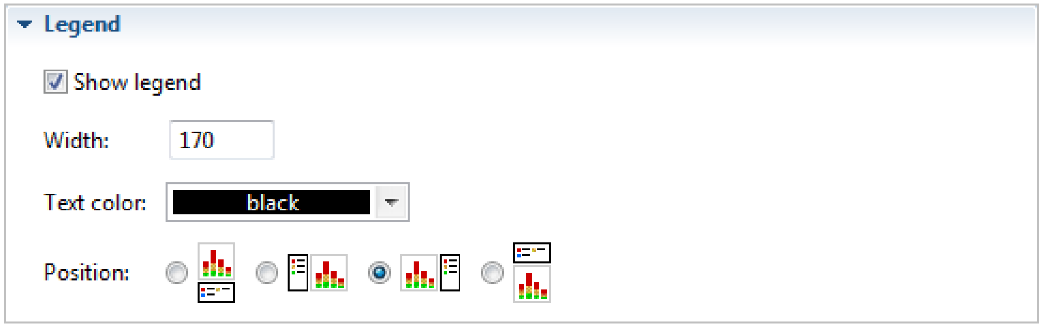
Data

Title: ATM utilization

Color: mediumTurquoise

Value: ATM.statsUtilization.mean()

5. Change the position of the chart's legend in the **Legend** properties section.



Legend

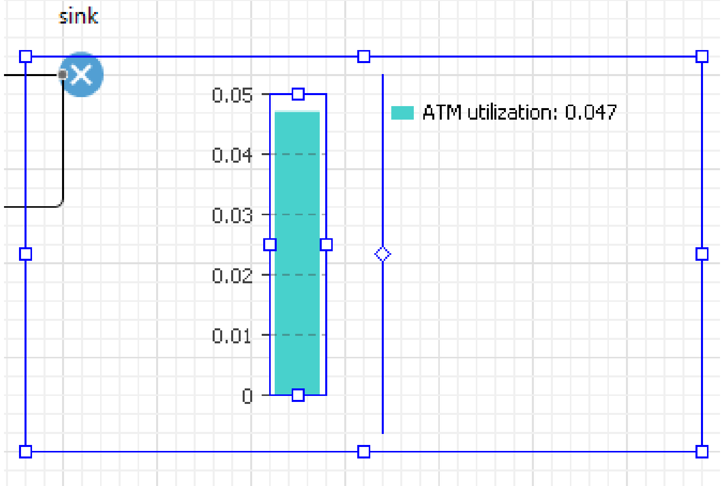
☑ Show legend

Width: 170

Text color: black

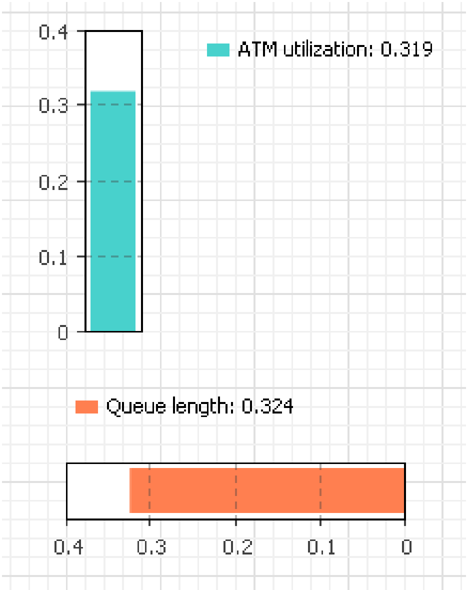
Position: ○ ○ ● ○

6. Then resize the chart as shown in the figure below:

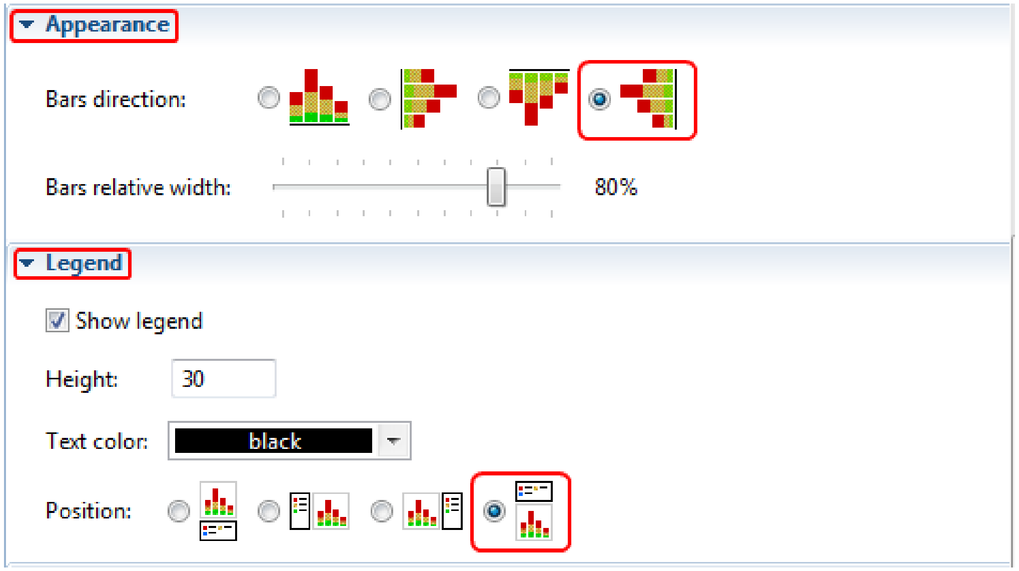


Add a bar chart to indicate mean queue length

1. Add one more bar chart in the same way. Resize it to look like the one in the figure.



2. Open the **Appearance** section of the **Properties** view and choose the last option from the **Bars Direction** choice to make bars grow to the left and also change the position of the chart's legend in the section **Legend** (like it is shown in the figure below).



Appearance

Bars direction: ○ ○ ○ ●

Bars relative width: 80%

Legend

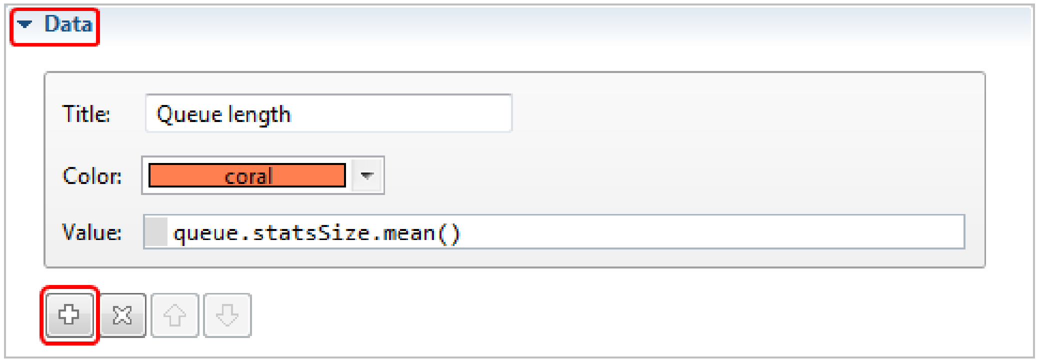
☑ Show legend

Height: 30

Text color: black

Position: ○ ○ ○ ●

3. Add a data item to be displayed by the chart. Set **Title**: *Queue length* and **Value**: `queue.statsSize.mean()`
   Here `statsSize` is the data set of type **StatisticsContinuous** that collects the statistics on the **Queue** size.

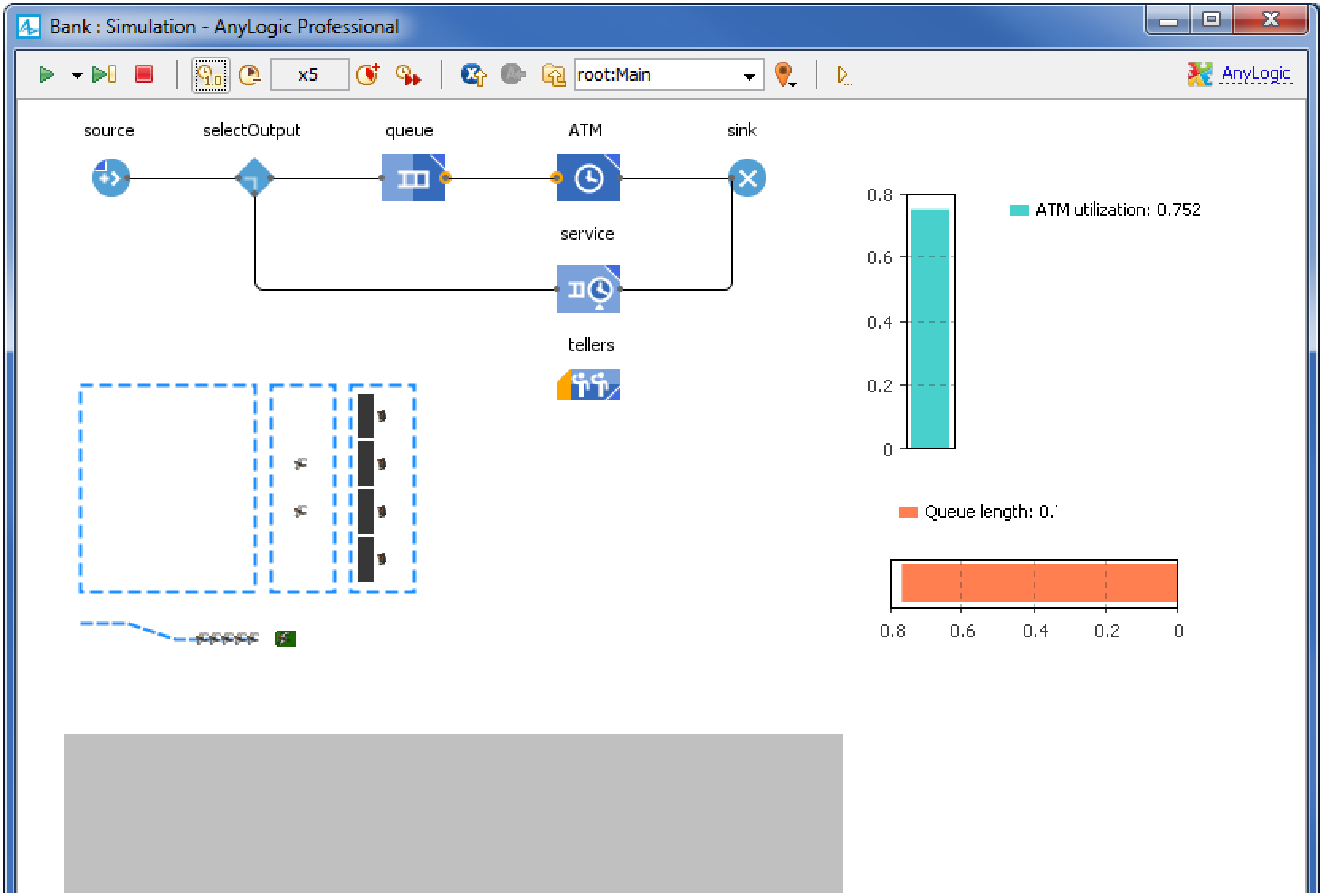


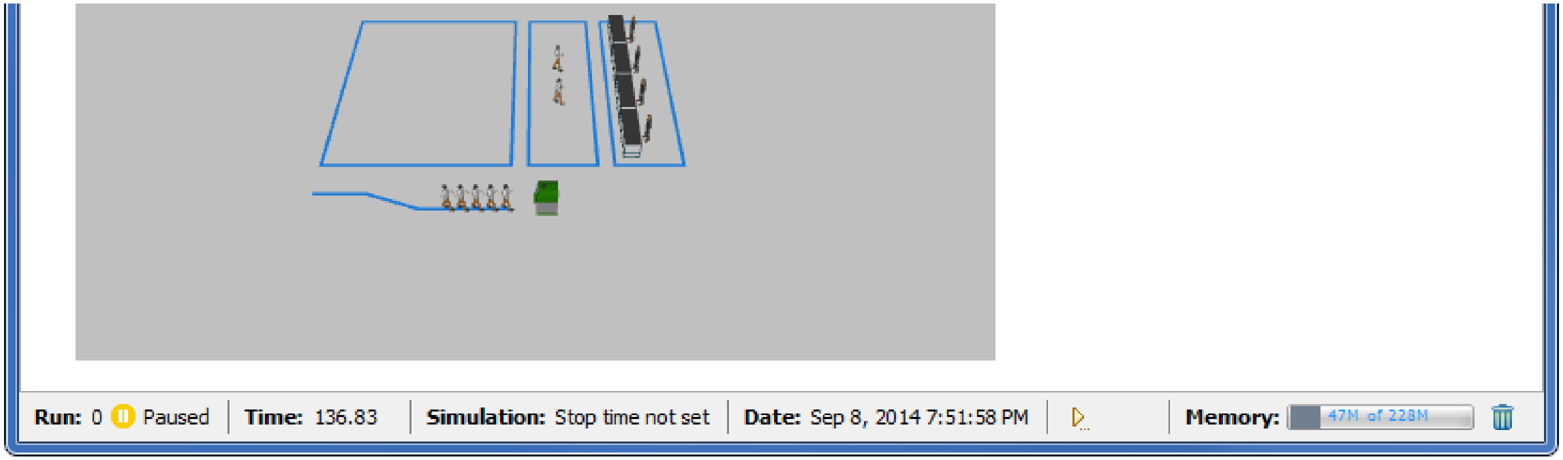
Title: Queue length

Color: coral

Value: queue.statsSize.mean()

Run the model and observe the ATM utilization and mean queue length with just created charts.





Run: 0 Paused | Time: 136.83 | Simulation: Stop time not set | Date: Sep 8, 2014 7:51:58 PM | Memory: 47M of 228M

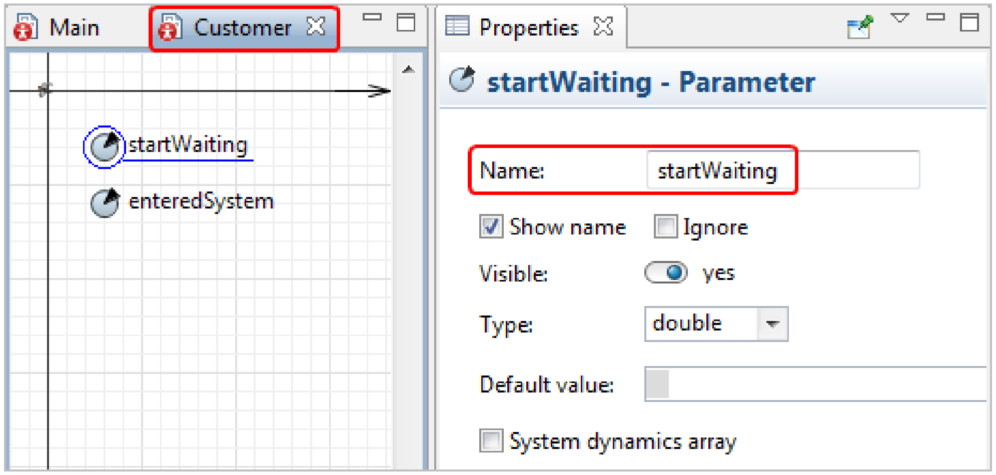
Collecting customer time statistics

We want to know how much time customer spends waiting in ATM queue and the whole time he spends in the bank. We will collect time statistics using AnyLogic analysis data objects and observe the resulting time distributions using histograms. We will use the agent type `Customer` we have created on the step 2.

First we need to add two parameters into our model.

Add parameters

1. Switch to the **Projects** view. Double-click the agent type `Customer` to open its diagram. We need to create parameteres for the agent type `Customer` if we want to collect customer statistics.
2. Open the **Agent** palette of the **Palette** view.
3. Drag the element **Parameter** into the diagram of `Customer`.
4. Leave the type `double` as it stands by default and name the parameters *startWaiting* and *enteredSystem*.



startWaiting - Parameter

Name: startWaiting

☑ Show name □ Ignore

Visible: yes

Type: double

Default value:

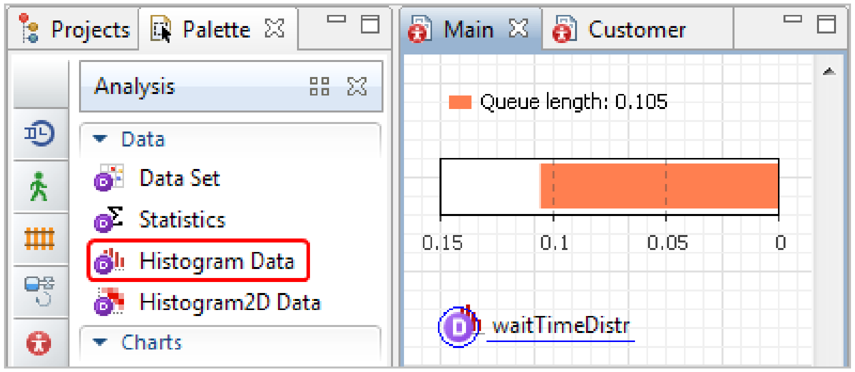
□ System dynamics array

5. We will continue developing our model in the `Main` diagram.

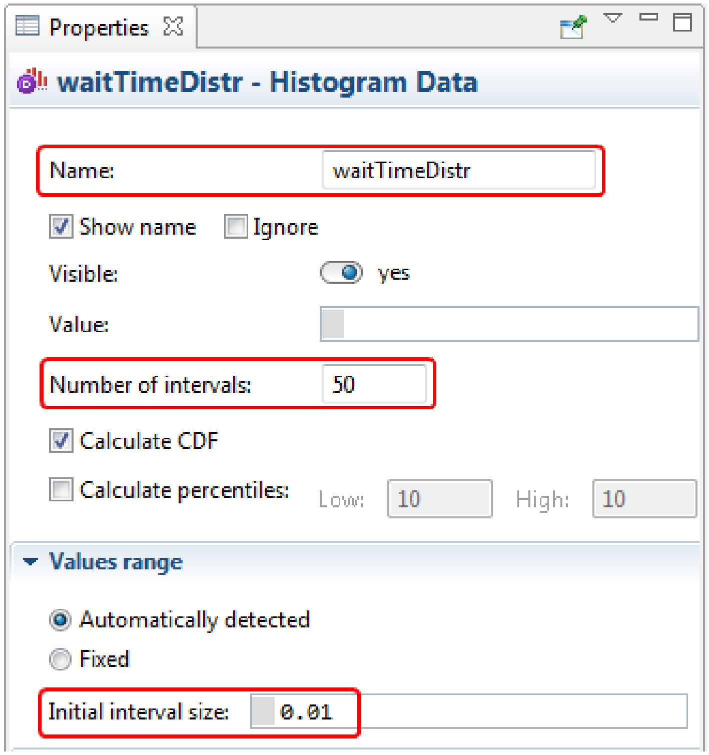
Add histogram data objects to store statistics on customer's waiting time and time in system. Histogram data objects support standard statistical analysis on the data values being added (calculate mean, minimum, maximum, deviation, variance and mean confidence interval).

Add histogram data objects to collect time statistics

1. To add histogram data object on the diagram, drag the **Histogram Data** from the **Analysis** palette of the **Palette** onto the `Main` diagram.



2. Set up the properties of the element.
   - Change the **Name** to *waitTimeDistr*.
   - Set the **Number of intervals** equal to *50*.
   - Set the **Initial interval size**: *0.01*.



Properties

waitTimeDistr - Histogram Data

Name: waitTimeDistr

☑ Show name □ Ignore

Visible: yes

Value:

Number of intervals: 50

☑ Calculate CDF

□ Calculate percentiles: Low: 10 High: 10

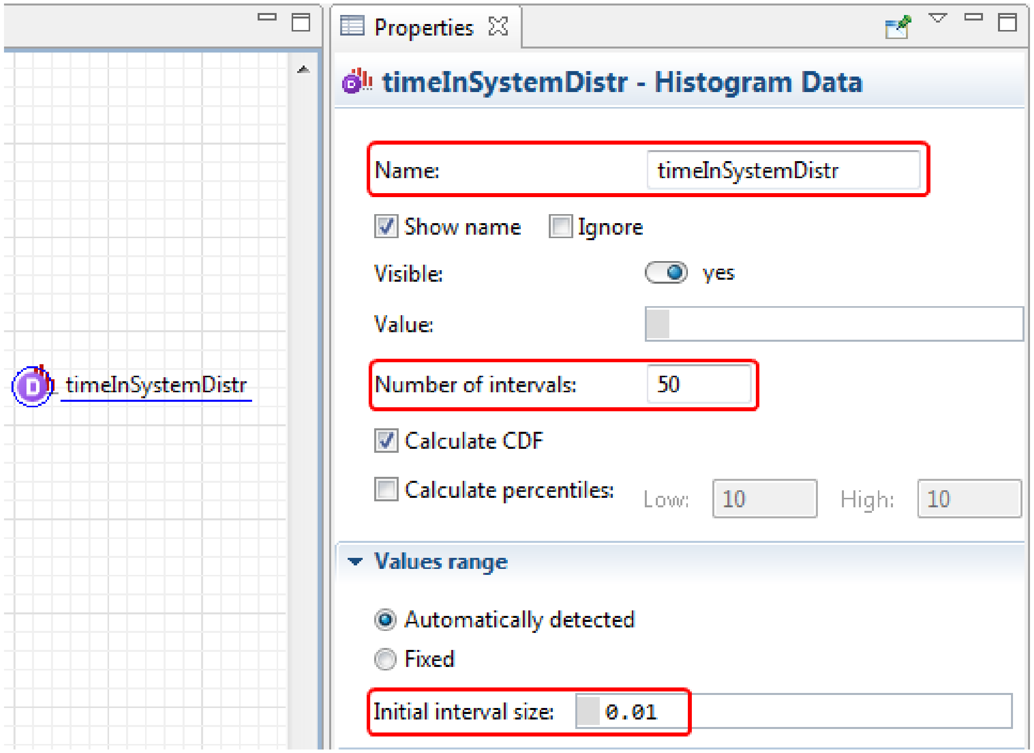
Values range

● Automatically detected

○ Fixed

Initial interval size: `0.01`

3. Create one more histogram data object. `Ctrl`+drag histogram data object to create its copy. Change the **Name** to *timeInSystemDistr*.



timeInSystemDistr - Histogram Data

Name: timeInSystemDistr

☑ Show name □ Ignore

Visible: yes

Value:

Number of intervals: 50

☑ Calculate CDF

□ Calculate percentiles: Low: 10 High: 10

Values range

● Automatically detected

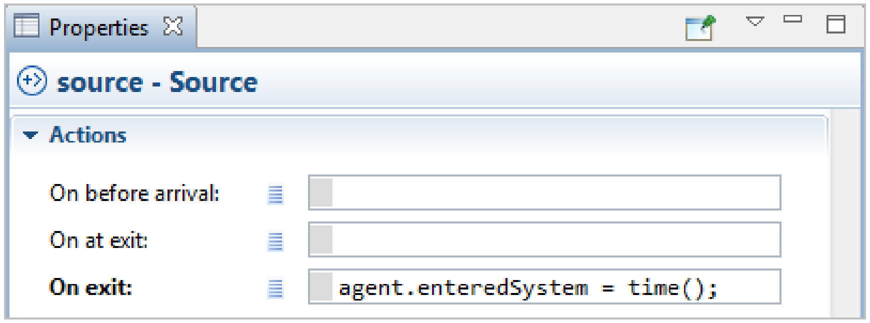
○ Fixed

Initial interval size: 0.01

Now we will modify properties of our flowchart objects.

Modify the properties of the flowchart objects

1. Modify *source* properties:
   - Make sure the agent type `Customer` is specified as **New agent**. This object should continue generating agents of our `Customer` type.
   - Type `agent.enteredSystem = time();` in **On exit** action, located in **Actions** section. This code stores the time when a customer was generated in the `Customer`'s variable `enteredSystem`. The `time()` function returns the current model time value.



source - Source

Actions

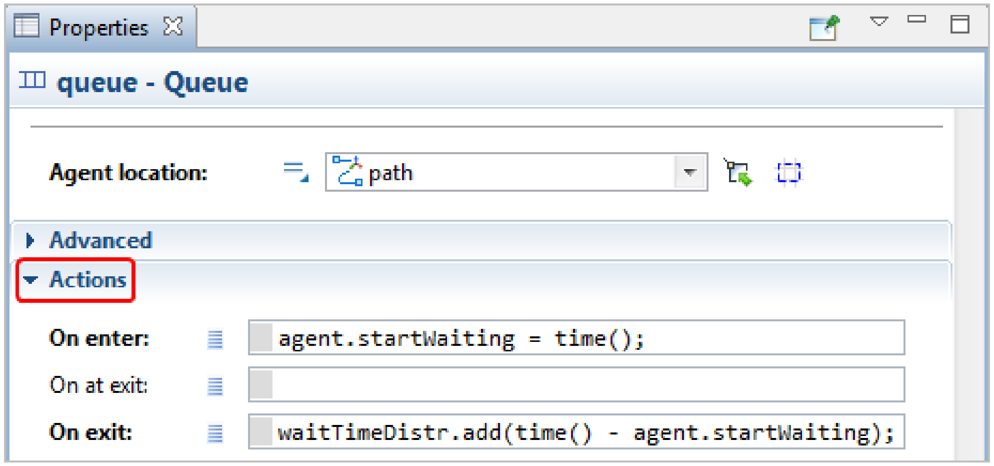
On before arrival:

On at exit:

On exit: `agent.enteredSystem = time();`

2. Modify *queue* properties:

- Type `agent.startWaiting = time();` in **On enter** action, located in **Actions** section. This code stores the time when a customer started waiting in the queue in the `Customer`'s variable `startWaiting`.
- Type `waitTimeDistr.add(time() - agent.startWaiting);` in **On exit**. This code adds waiting time of the customer to the *waitTimeDistr* histogram data object.



queue - Queue

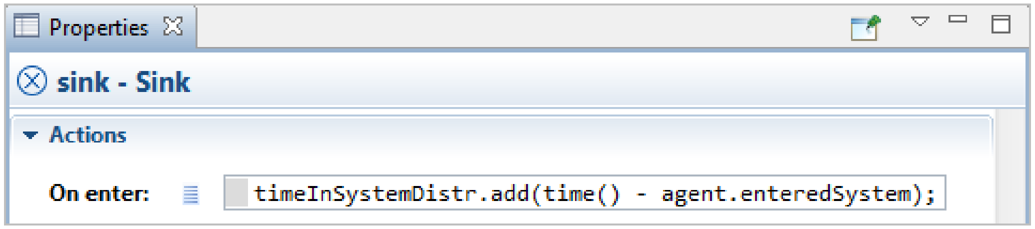
Agent location: path

On enter: `agent.startWaiting = time();`

On at exit:

On exit: `waitTimeDistr.add(time() - agent.startWaiting);`

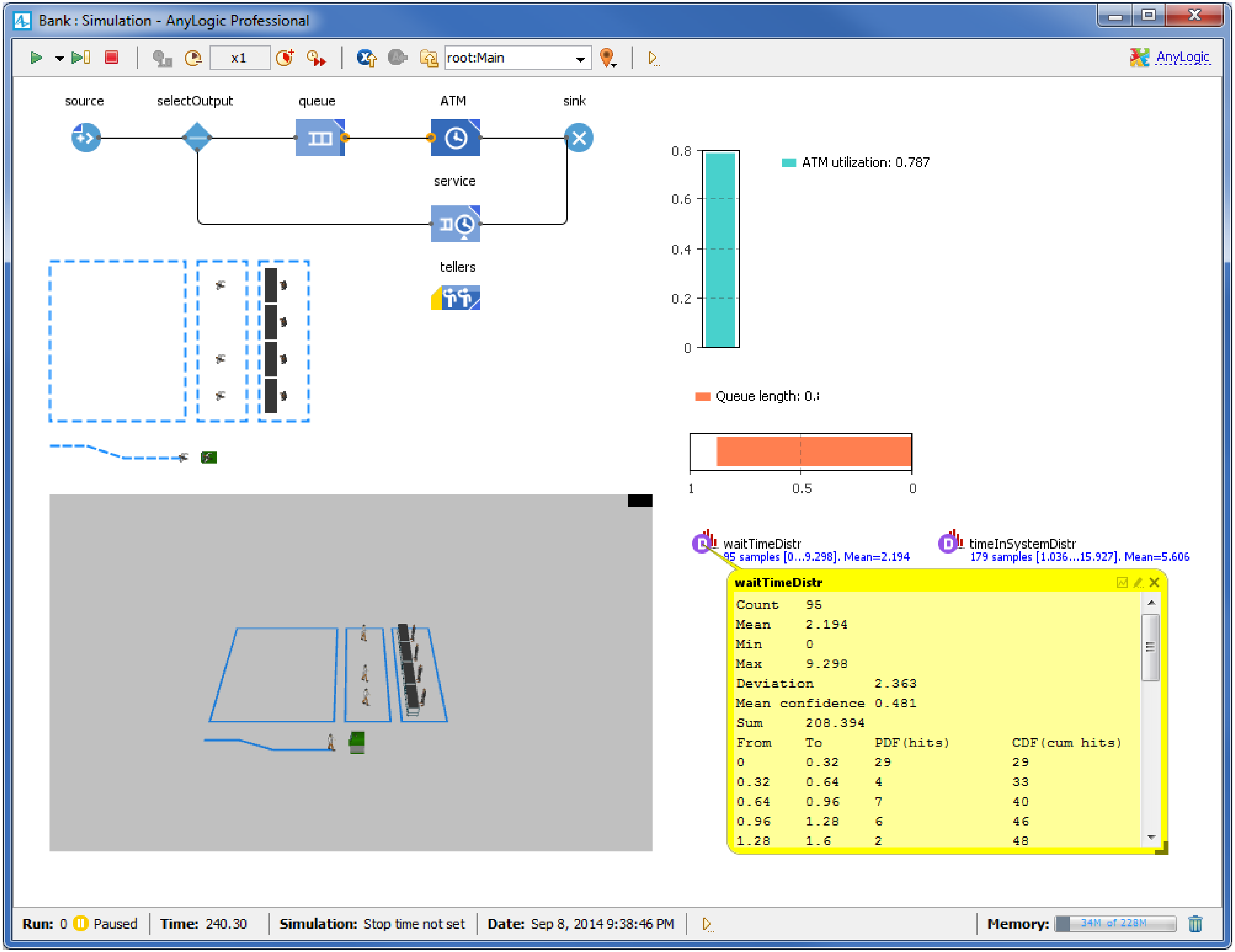
3. Modify *sink* properties:
    - Type `timeInSystemDistr.add(time() - agent.enteredSystem);` in **On enter** action, located in **Actions** section. This code adds the whole time the customer spent in the bank to the *timeInSystemDistr* histogram data object.



sink - Sink

On enter: `timeInSystemDistr.add(time() - agent.enteredSystem);`

Run the model and view the statistics using inspect window of the data set. Open inspect window for data set by clicking on it. Here you can see standard statistical analysis on the data values being added to this data object.



waitTimeDistr

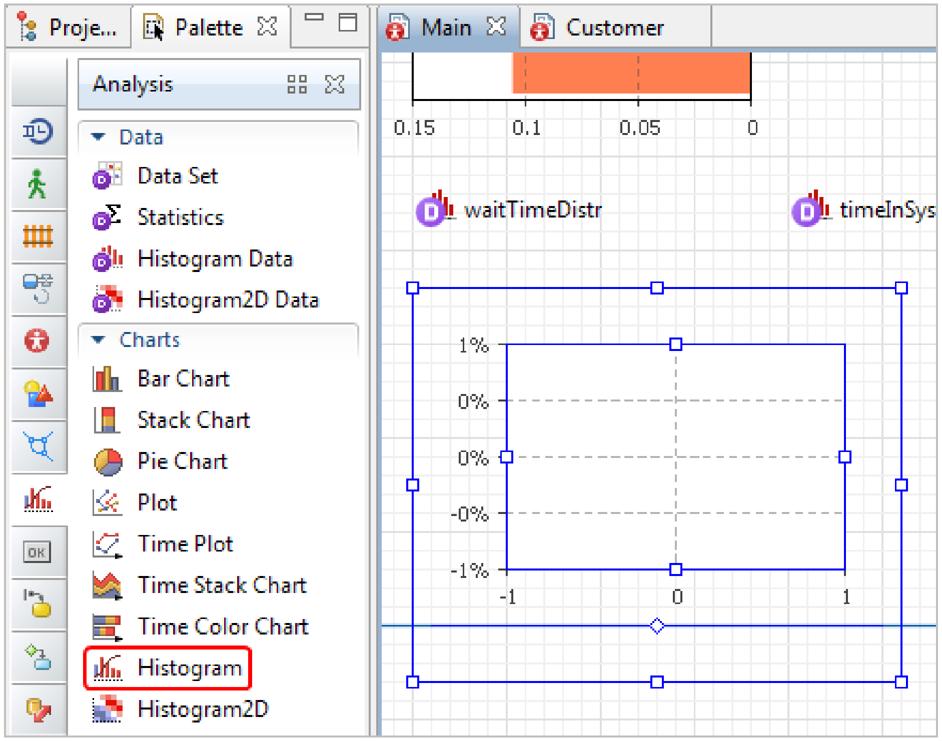
```
Count    95
Mean     2.194
Min      0
Max      9.298
Deviation       2.363
Mean confidence 0.481
Sum      208.394
From     To      PDF(hits)     CDF(cum hits)
0        0.32    29            29
0.32     0.64    4             33
0.64     0.96    7             40
0.96     1.28    6             46
1.28     1.6     2             48
```

Run: 0 Paused | **Time:** 240.30 | **Simulation:** Stop time not set | **Date:** Sep 8, 2014 9:38:46 PM | **Memory:** 34M of 228M

Now we want to display the collected statistics using standard histograms.

Add two histograms to display distributions of customer's waiting time and time in system

1. To add histogram on the diagram, drag the **Histogram** element from the **Analysis** palette of the **Palette** onto the diagram where you want to place the histogram. Resize it if needed.



2. Define the data object to be displayed by this histogram. Click the **Add histogram data** button in **Data** section and change the **Title** to *Waiting time distribution*. Specify the **Histogram** to be displayed: `waitTimeDistr`

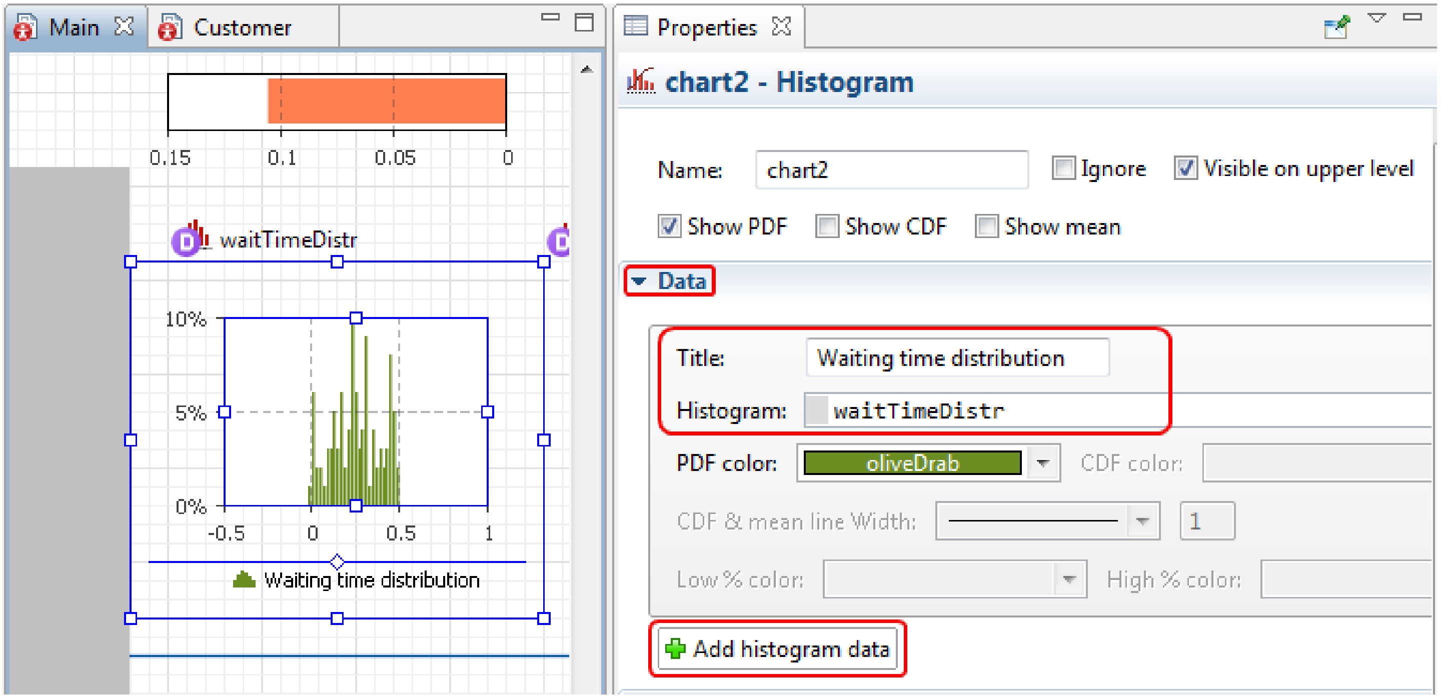


chart2 - Histogram

Name: chart2 ☐ Ignore ☑ Visible on upper level

☑ Show PDF ☐ Show CDF ☐ Show mean

Data

Title: Waiting time distribution

Histogram: waitTimeDistr

PDF color: oliveDrab CDF color:

CDF & mean line Width: 1

Low % color: High % color:

3. Add one more histogram below the existing one.
   Change the **Title** of the displayed data to *Time in system distribution*
   Choose the data object to be displayed: `timeInSystemDistr`

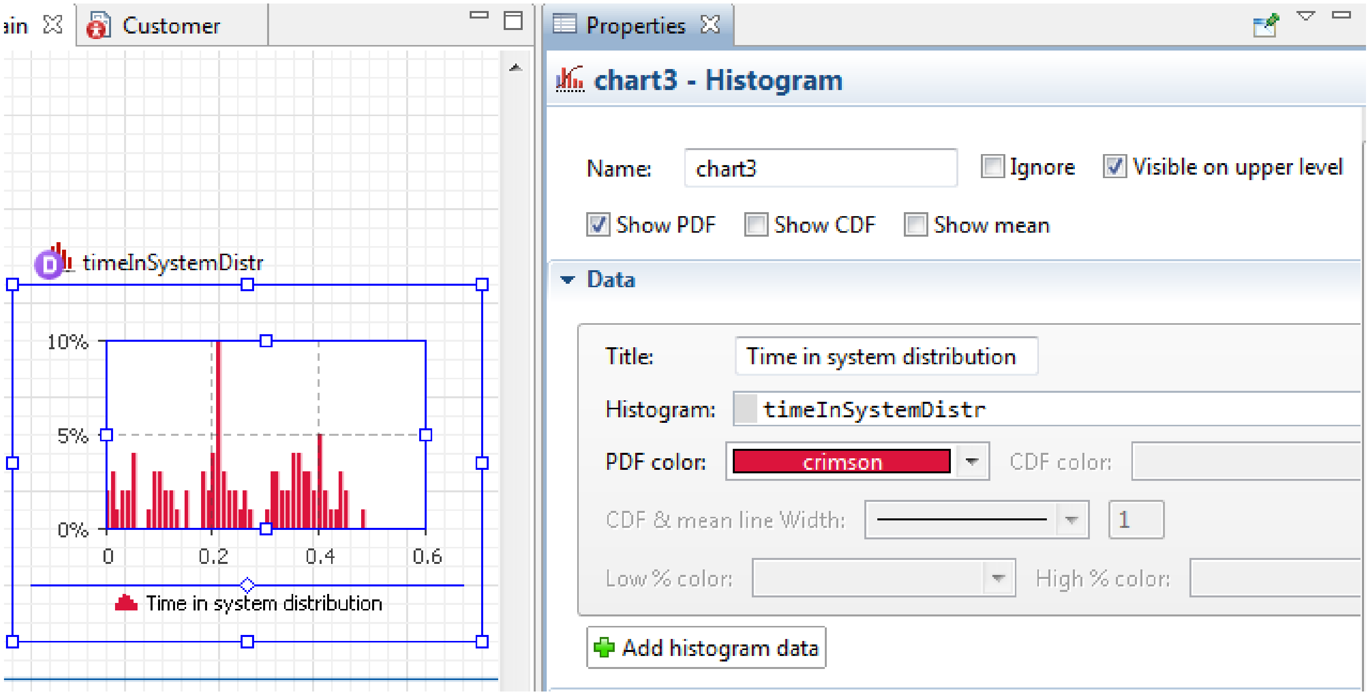


chart3 - Histogram

Name: chart3 □ Ignore ☑ Visible on upper level

☑ Show PDF □ Show CDF □ Show mean

Data

Title: Time in system distribution

Histogram: timeInSystemDistr

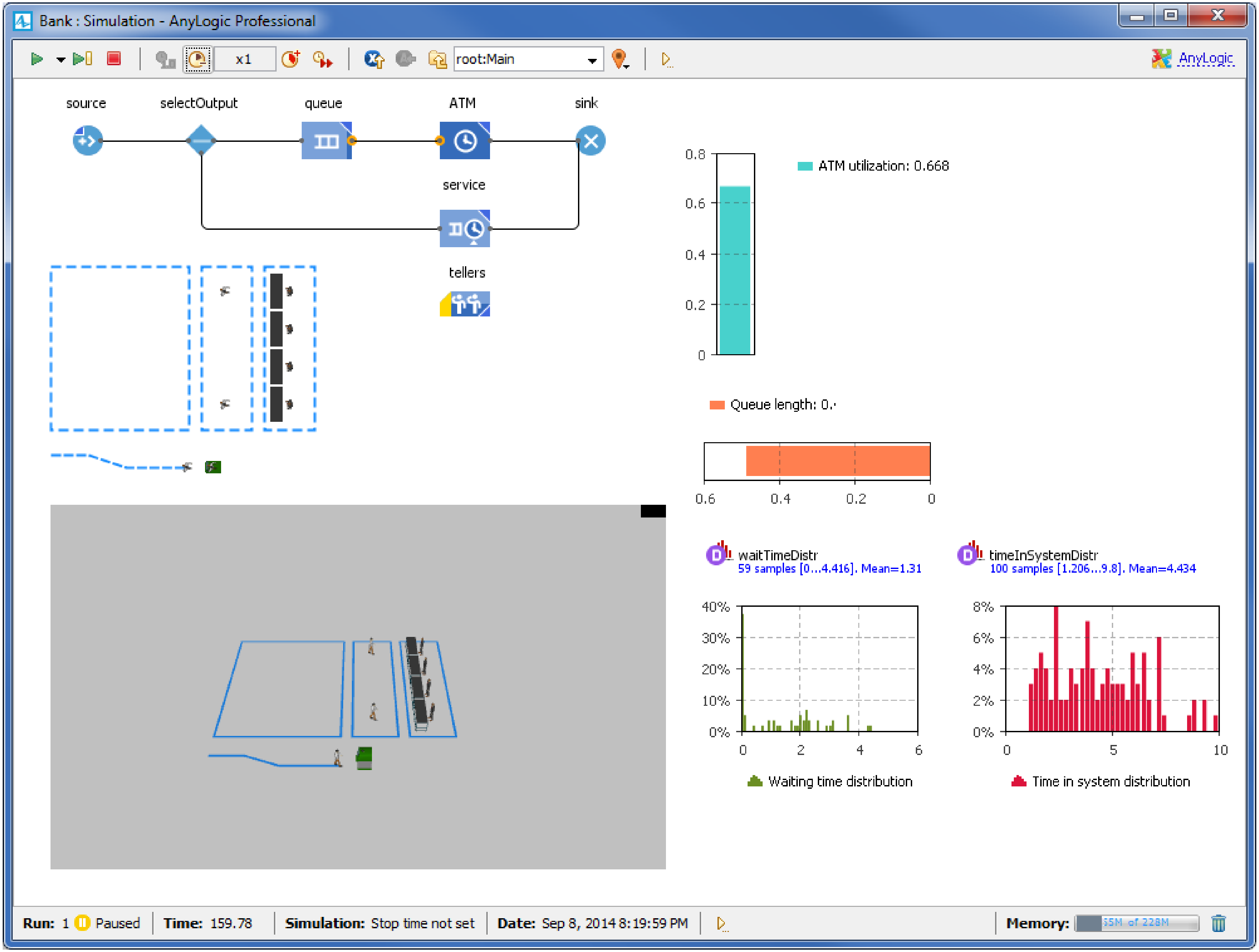
PDF color: crimson CDF color:

CDF & mean line Width: 1

Low % color: High % color:

Add histogram data

Run the model. Set virtual time mode and observe distribution of customer's time in system and waiting time.



waitTimeDistr 59 samples [0...4.416]. Mean=1.31

timeInSystemDistr 100 samples [1.206...9.8]. Mean=4.434

Run: 1 Paused **Time:** 159.78 **Simulation:** Stop time not set **Date:** Sep 8, 2014 8:19:59 PM **Memory:** 55M of 228M

Reference model: Bank - Phase 4

« Phase 3, Adding tellers