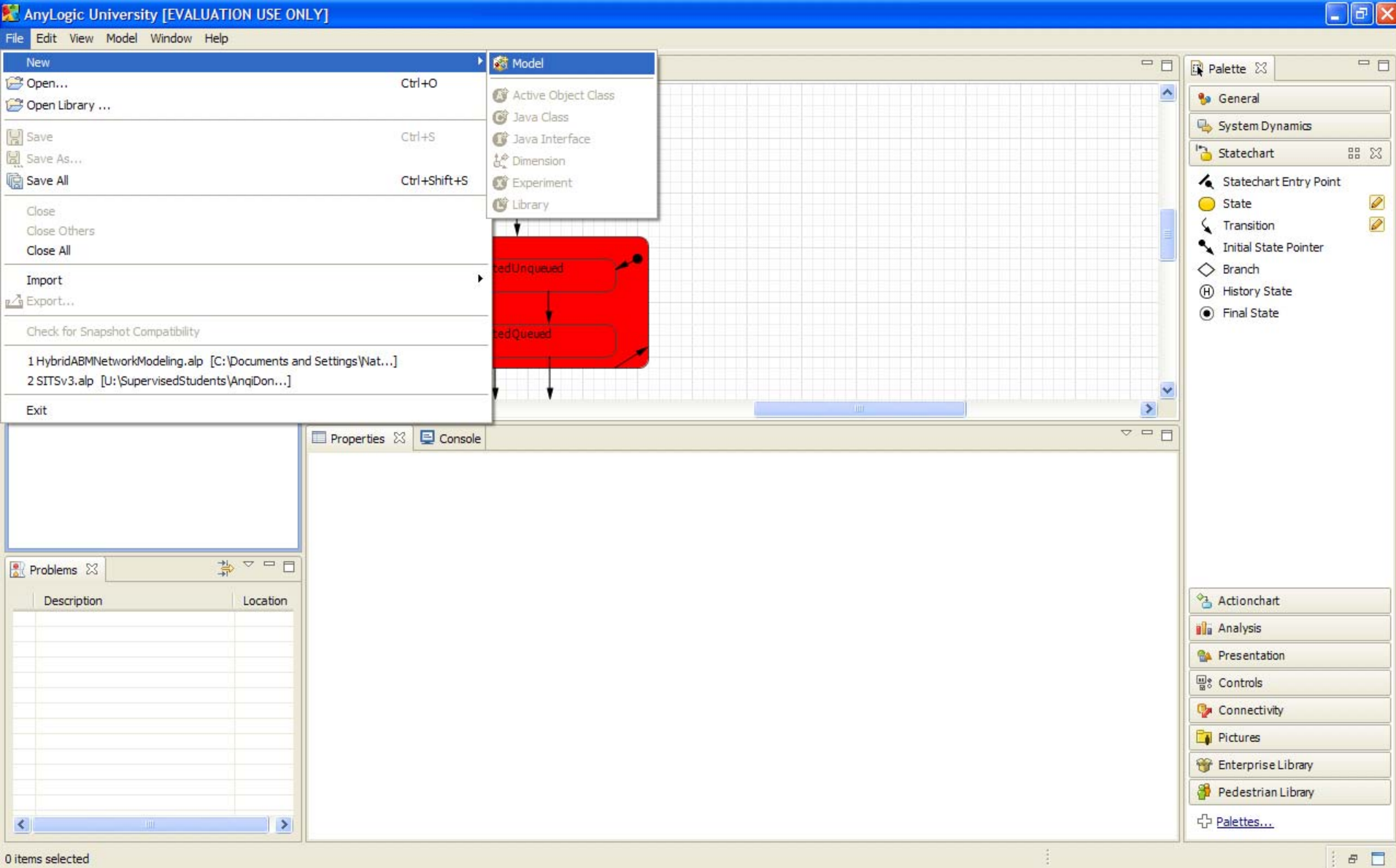


Creating a Hybrid Agent-Based and Network (Patient-Flow) Model

Nathaniel Osgood

11-7-2009

Request to Create a Model



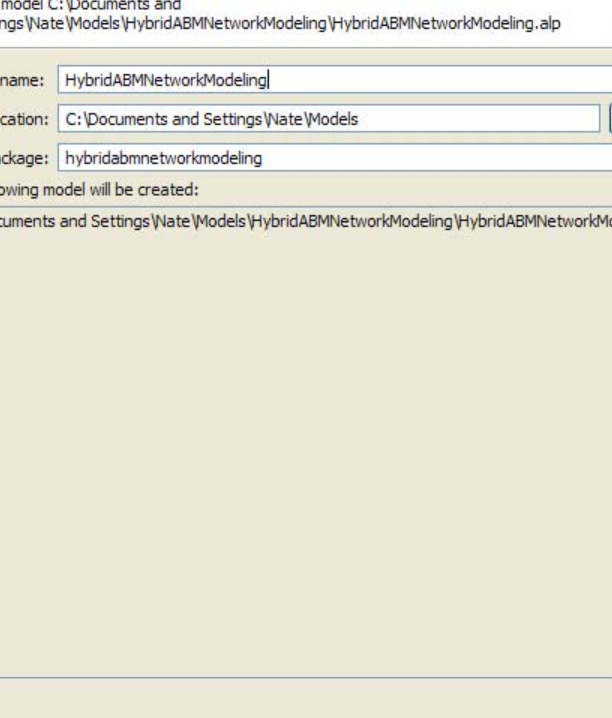
The screenshot displays the AnyLogic software interface. A 'New Model' dialog box is open in the center, prompting for the following information:

- Model name:** HybridABMNetworkModeling
- Location:** C:\Documents and Settings\Nate\Models (with a 'Browse...' button)
- Java Package:** hybridabmnetworkmodeling

Below these fields, it states: 'The following model will be created: C:\Documents and Settings\Nate\Models\HybridABMNetworkModeling\HybridABMNetworkModeling.alp'. At the bottom of the dialog are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

In the background, the main interface is visible:

- Project Tree (Left):** Shows a project named 'SITsV3' with components like Main, Parameters, Plain Variables, Functions (including 'PickRandomPerson'), Events (including 'event'), Environments, Embedded Objects, Analysis Data, Presentation, and a 'Person' component.
- Palette (Right):** Contains various modeling elements categorized under 'General', 'System Dynamics', 'Statechart' (including Statechart Entry Point, State, Transition, Initial State Pointer, Branch, History State, Final State), 'Actionchart', 'Analysis', 'Presentation', 'Controls', 'Connectivity', 'Pictures', 'Enterprise Library', 'Pedestrian Library', and 'Palettes...'.
- Problems Window (Bottom Left):** A table with columns 'Description' and 'Location'.
- Status Bar (Bottom):** Indicates '0 items selected'.



New Model

The model C:\Documents and Settings\Nate\Models\HybridABMNetworkModeling\HybridABMNetworkModeling.alp

Model name: HybridABMNetworkModeling

Location: C:\Documents and Settings\Nate\Models Browse...

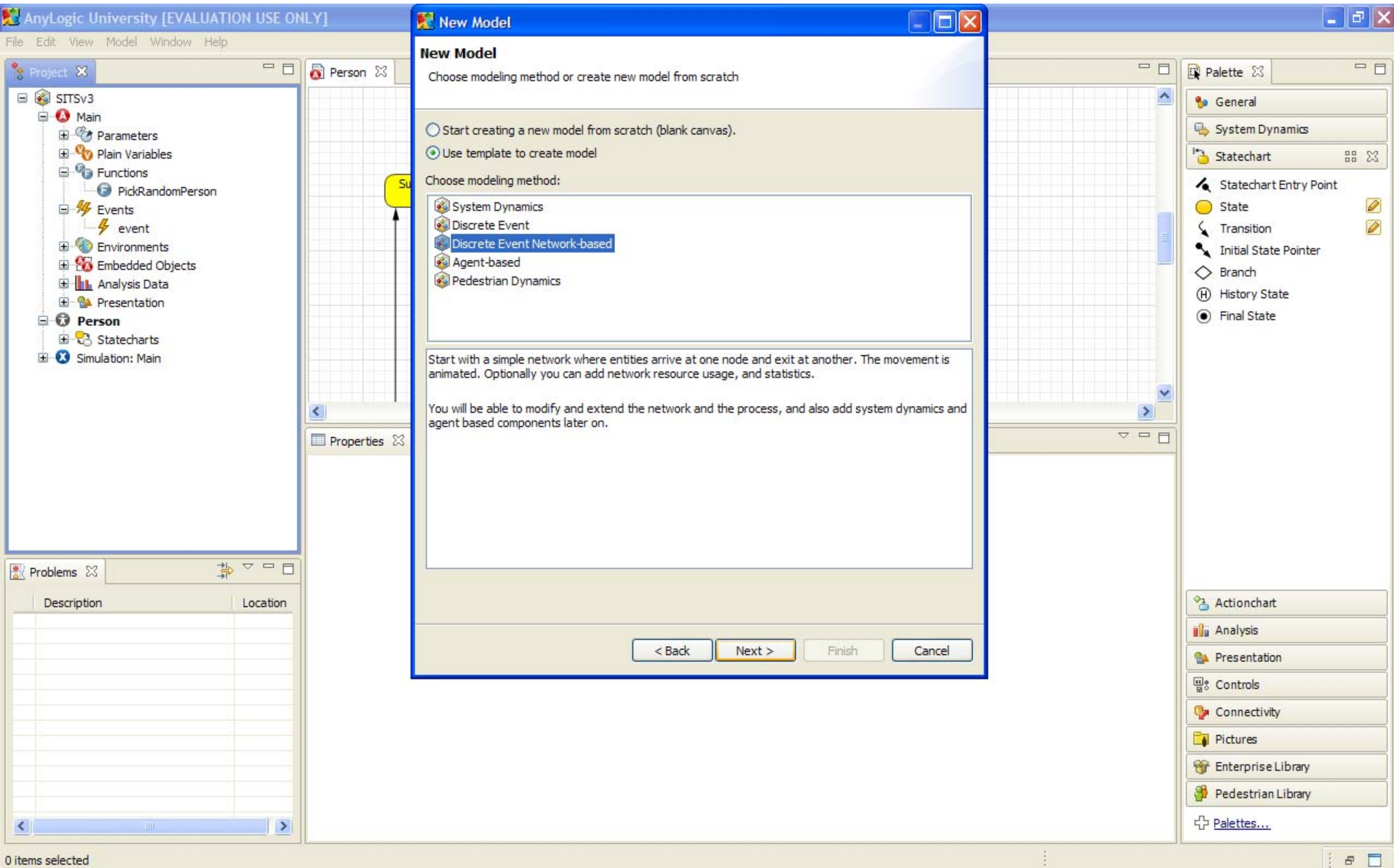
Java Package: hybridabmnetworkmodeling

The following model will be created:

C:\Documents and Settings\Nate\Models\HybridABMNetworkModeling\HybridABMNetworkModeling.alp

< Back **Next >** Finish Cancel

Indicate that this is to be a Network Model



Create a Model without Statistics

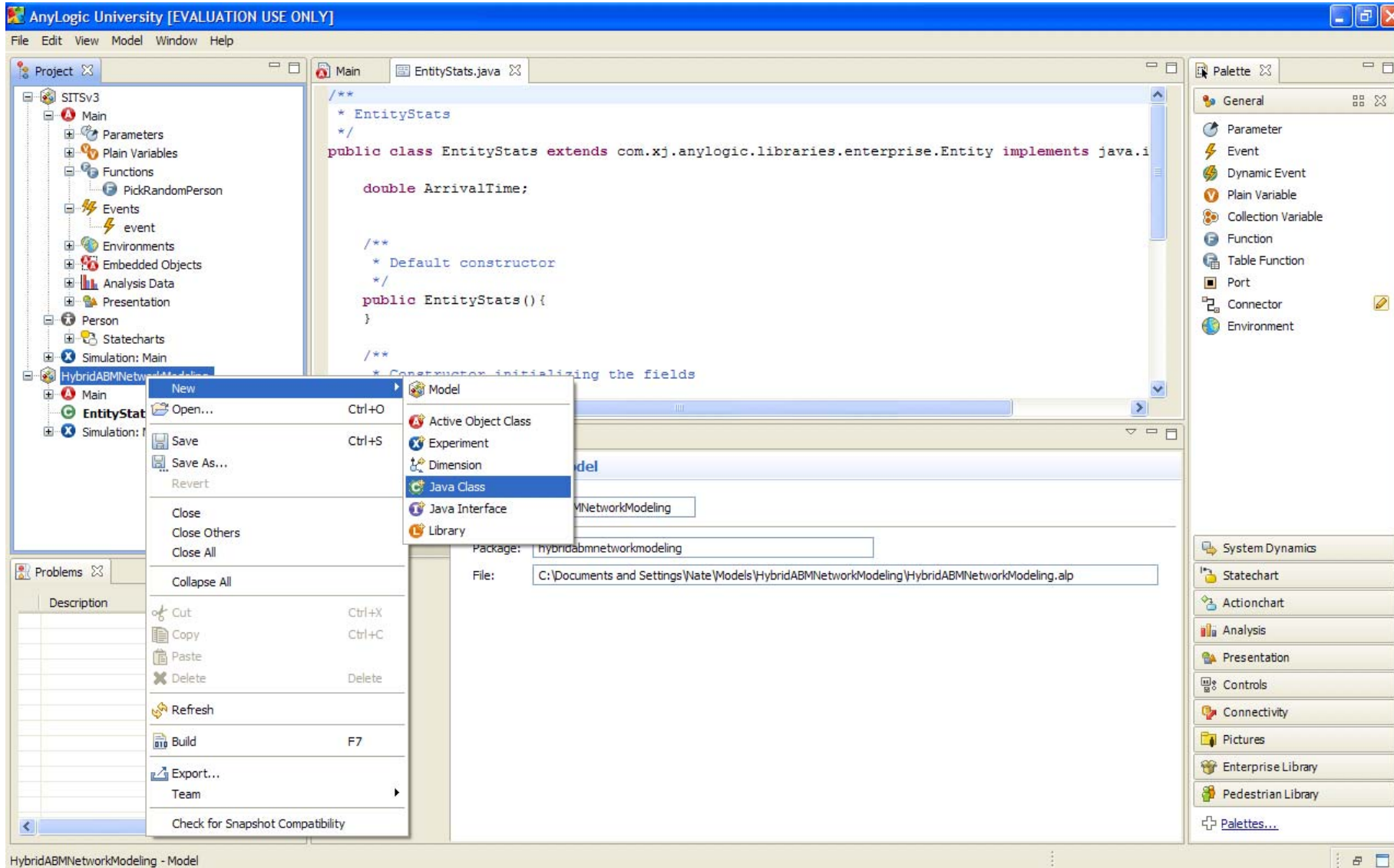
The screenshot shows the AnyLogic University [EVALUATION USE ONLY] interface. The 'New Model' dialog box is open, displaying the 'Create Discrete Event Network-based Model' options. The 'Specify model options' section has three checkboxes: 'Use resources' (checked), 'Resource utilization statistics' (checked), and 'Add time in system histogram' (unchecked). Below the checkboxes is a diagram illustrating the flow of a discrete event network-based model. The diagram shows a sequence of nodes: 'source' (yellow circle), 'networkEnter' (blue square), 'networkMoveTo' (blue square), 'networkSeize' (blue square), 'delay' (yellow circle), 'networkRelease' (blue square), 'networkMoveTo1' (blue square), 'networkExit' (blue square), and 'sink' (yellow circle). A 'networkResourcePool' (blue square) is also shown. The flow is: source → networkEnter → networkMoveTo → networkSeize → delay → networkRelease → networkMoveTo1 → networkExit → sink. A 'network' node (blue square) is connected to the 'networkResourcePool'.

Entities arrive at an entry node, move to another node, request a resource unit, wait until it arrives, use resource for a while, release it, and proceed to the exit node.

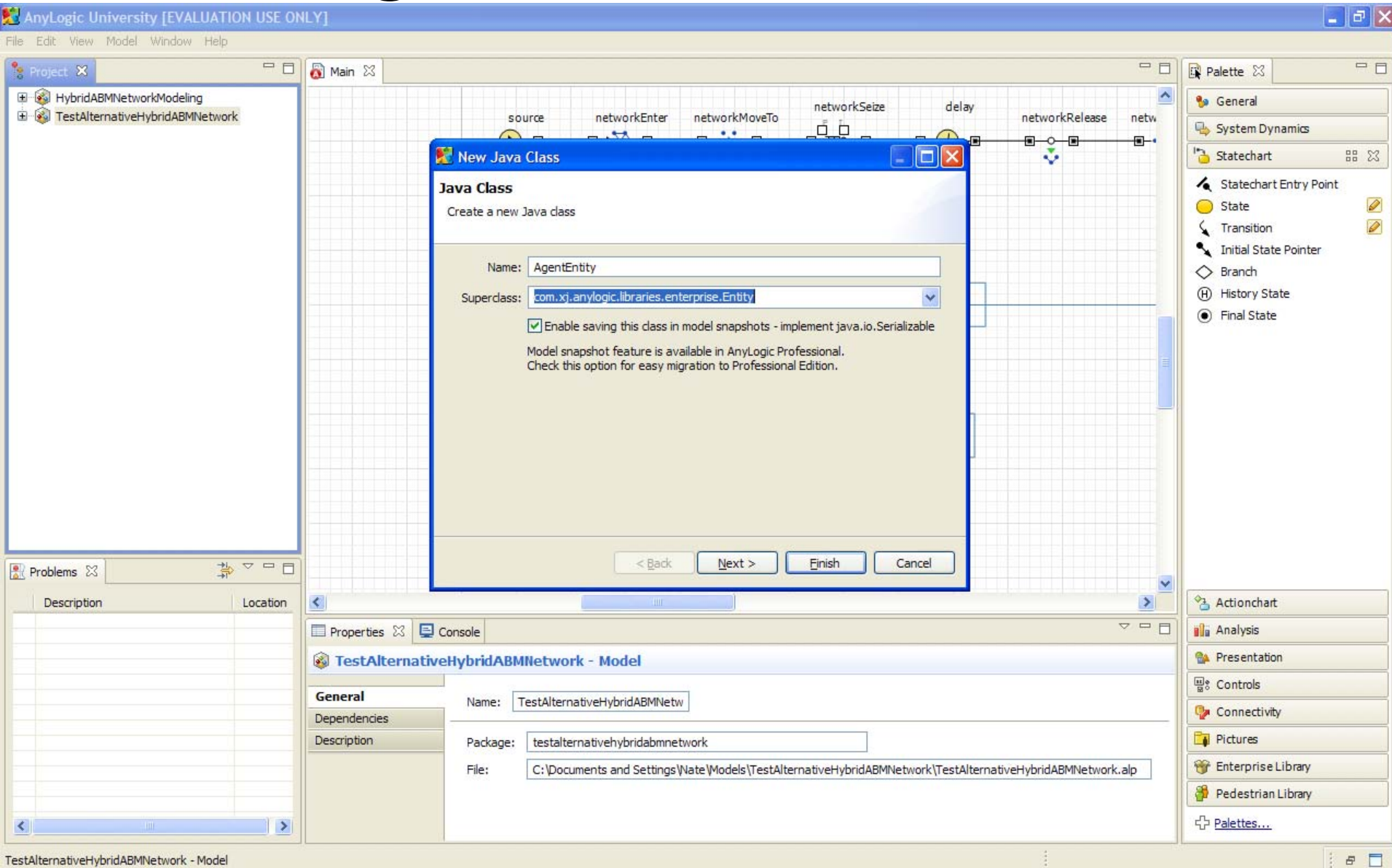
- A bar chart displaying the resource utilization is included.

The dialog box has buttons for '< Back', 'Next >', 'Finish', and 'Cancel'. The background shows the AnyLogic project environment with a 'Project' window containing 'HybridABMNetworkModeling', a 'Problems' window, and a 'Palette' window with various modeling elements like 'Statechart Entry Point', 'State', 'Transition', 'Initial State Pointer', 'Branch', 'History State', 'Final State', 'Actionchart', 'Analysis', 'Presentation', 'Controls', 'Connectivity', 'Pictures', 'Enterprise Library', 'Pedestrian Library', and 'Palettes...'. The status bar at the bottom indicates '0 items selected'.

Add a Class



Giving the New Class a Name



Resulting class

The screenshot displays the AnyLogic University software interface. The main window shows the Java code for the `AgentEntity` class, which extends `com.xj.anylogic.libraries.enterprise.Entity` and implements `java.i`. The code includes a default constructor, an overridden `toString()` method, and a `serialVersionUID`.

Project Explorer (Left): Shows the project structure for `SITSV3`. The `AgentEntity` class is highlighted under the `Simulation: Main` package.

Code Editor (Center): Displays the following Java code:

```
/**
 * AgentEntity
 */
public class AgentEntity extends com.xj.anylogic.libraries.enterprise.Entity implements java.i

/**
 * Default constructor
 */
public AgentEntity(){
}

@Override
public String toString() {
    return super.toString();
}

/**
 * This number is here for model snapshot storing purpose<br>
 * It needs to be changed when this class gets changed
 */
private static final long serialVersionUID = 1L;
}
```

Palette (Right): Lists various components available for the model, including Parameter, Event, Dynamic Event, Plain Variable, Collection Variable, Function, Table Function, Port, Connector, and Environment.

System Dynamics (Bottom Right): Lists various system dynamics components, including Statechart, Actionchart, Analysis, Presentation, Controls, Connectivity, Pictures, Enterprise Library, Pedestrian Library, and Palettes...

Properties and Console (Bottom): The Properties panel is empty. The Console panel shows the output of the `toString()` method, displaying the class name `AgentEntity`.

Make the AgentEntity capable of carrying around & curing your Agents

The screenshot displays the AnyLogic University software interface. The main window shows the code for the `AgentEntity` class, which extends `com.xj.anylogic.libraries.enterprise.Entity` and implements `java.i`. The code includes a private field `associatedPerson` of type `Person`, a default constructor, a constructor taking a `Person` parameter, a `Cure()` method that sends a "Cured!" message to the associated person, and a `toString()` method that returns a string representation of the entity. The code also includes a static final `serialVersionUID`.

The interface includes a Project Explorer on the left showing the model structure, a Palette on the right with various components, and a Console at the bottom for output.

```
/**
 * AgentEntity
 */
public class AgentEntity extends com.xj.anylogic.libraries.enterprise.Entity implements java.i
{
    private Person associatedPerson = null;

    /**
     * Default constructor
     */
    public AgentEntity() {
    }

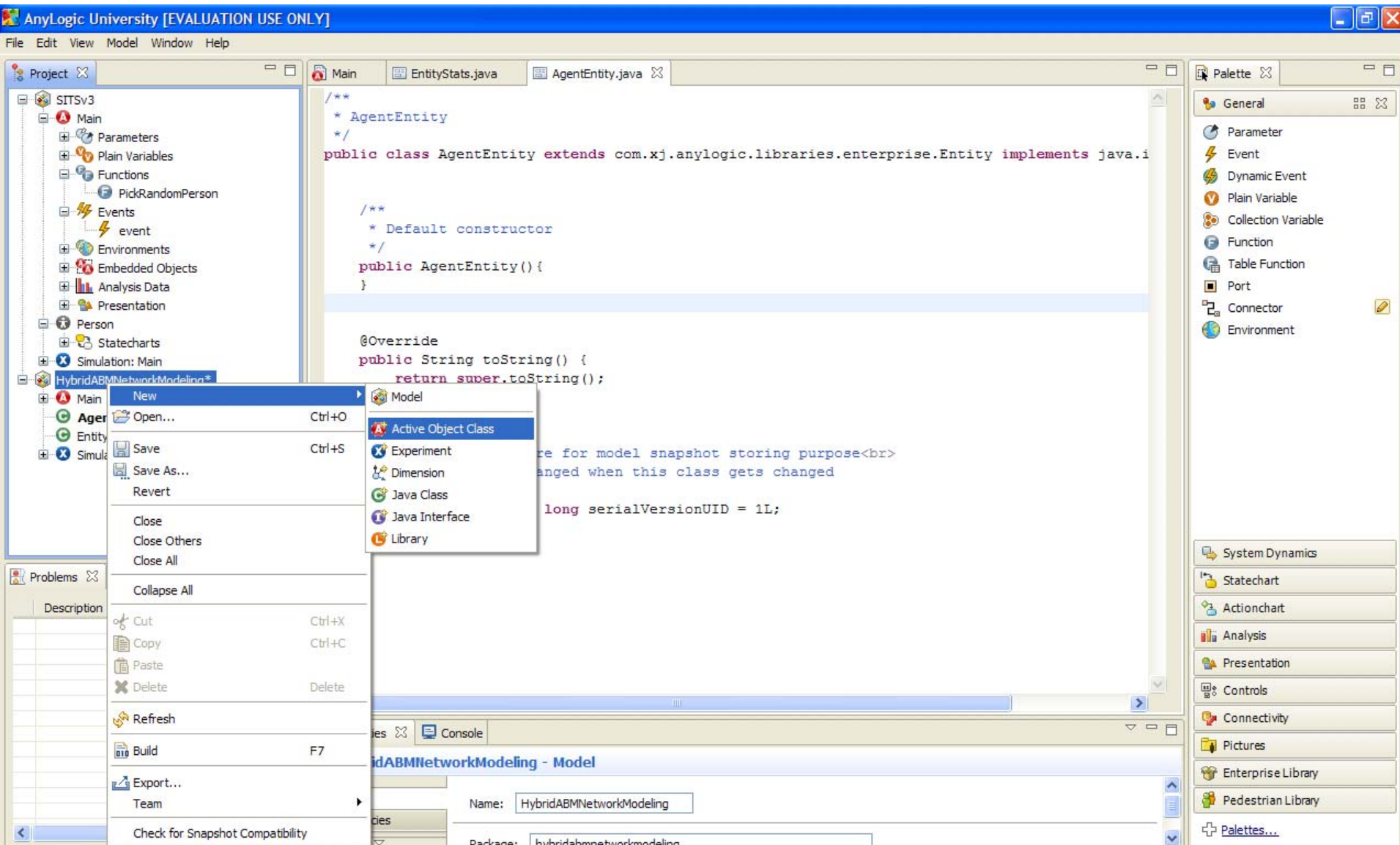
    public AgentEntity(Person p)
    {
        associatedPerson = p;
    }

    public void Cure()
    {
        associatedPerson.send("Cured!", associatedPerson);
    }

    @Override
    public String toString()
    {
        return "Entity for agent " + associatedPerson;
    }

    /**
     * This number is here for model snapshot storing purpose<br>
     * It needs to be changed when this class gets changed
     */
    private static final long serialVersionUID = 1L;
}
```

Add Person Active Object class



Indicate that the Person is an Agent

The screenshot displays the AnyLogic University software interface, specifically the configuration window for the 'Person' entity. The interface is divided into several panels:

- Project Panel (Left):** Shows the project hierarchy for 'SITSV3'. The 'Person' entity is highlighted under the 'Main' folder.
- EntityStats.java, AgentEntity.java, Person (Tabs):** These tabs are visible at the top of the main workspace.
- Properties Panel (Bottom):** The 'Person - Active Object Class' properties are shown. The 'General' tab is selected, and the 'Agent' checkbox is checked, indicating that the 'Person' entity is configured as an agent.
- Console Panel (Bottom):** The 'Person - Active Object Class' console is visible, showing the 'Startup code' and 'Destroy code' fields.
- Palette (Right):** The 'General' palette is open, showing various components like Parameter, Event, Dynamic Event, Plain Variable, Collection Variable, Function, Table Function, Port, Connector, and Environment.
- System Dynamics (Right):** The 'System Dynamics' palette is open, showing components like Statechart, Actionchart, Analysis, Presentation, Controls, Connectivity, Pictures, Enterprise Library, and Pedestrian Library.

The status bar at the bottom indicates the selection of the 'Person' entity, with coordinates X=9, Y=539.

Build up an Agent with Desired Dynamics

AnyLogic University [EVALUATION USE ONLY]

File Edit View Model Window Help

Project

- Main
 - Parameters
 - Plain Variables
 - Functions
 - PickRandomPerson
 - Events
 - event
 - Environments
 - Embedded Objects
 - Analysis Data
 - Presentation
 - Person
 - Statecharts
 - Simulation: Main
 - HybridABMNetworkModeling*
- Main
 - Ports
 - Embedded Objects
 - Analysis Data
 - Presentation
 - Person
 - AgentEntity
 - EntityStats
 - Simulation: Main

Main EntityStats.java AgentEntity.java Person

statechart

Susceptible

Infected

Properties Console

transition1 - Transition

General

Name: transition1 ☐ Show name ☐ Ignore ☐ Public ☒ Show at runtime

Triggered by: Rate

Rate: 0.2

Action:

Guard:

Statechart Entry Point

State

Transition

Initial State Pointer

Branch

History State

Final State

Actionchart

Analysis

Presentation

Controls

Connectivity

Pictures

Enterprise Library

Pedestrian Library

Palettes...

transition1 - Transition

Selection

X=508, Y=239

Add an Environment

The screenshot displays the AnyLogic University software interface, which is used for building and simulating agent-based models. The interface is divided into several panes:

- Project Pane (Left):** Shows a hierarchical tree of the model's components. The 'Main' component is selected, and its sub-components are listed, including 'Parameters', 'Plain Variables', 'Functions', 'Events', 'Environments', 'Embedded Objects', 'Analysis Data', 'Presentation', 'Person', 'Statecharts', 'Simulation: Main', and 'HybridABMNetworkModeling*'. The 'Main' component is expanded, showing its sub-components: 'Ports', 'Environments', 'Embedded Objects', 'Analysis Data', 'Presentation', 'Person', 'AgentEntity', 'EntityStats', and 'Simulation: Main'.
- Main Canvas (Center):** Displays the model's visual representation. It includes a bar chart titled 'networkResourcePool' showing 'Resource utilization: 0.569'. A bar chart titled 'TimeInSystemHist' shows the distribution of 'Time in System' with a peak around 1.5. A small 'environment' icon is visible on the canvas.
- Properties Pane (Bottom):** Shows the properties of the selected 'environment' component. The 'General' tab is active, displaying the following settings:
 - Name: environment
 - Show name: ☒ (checked)
 - Ignore: ☐ (unchecked)
 - Public: ☐ (unchecked)
 - Show at runtime: ☒ (checked)
 - Enable steps: ☐ (unchecked)
 - Step duration (in model time units): [empty text box]
 - On before step: [empty text box]
 - On after step: [empty text box]
- Palette (Right):** A collection of icons for adding components to the model. The 'General' palette is visible, containing icons for Parameter, Event, Dynamic Event, Plain Variable, Collection Variable, Function, Table Function, Port, Connector, and Environment. The 'Environment' icon is highlighted.

The status bar at the bottom indicates the current component is 'environment - Environment', the action is 'Renaming', and the coordinates are 'X=688, Y=-84'.

Add a Population & Associate with Environment

AnyLogic University [EVALUATION USE ONLY]

File Edit View Model Window Help

Project

- Main
 - Parameters
 - Plain Variables
 - Functions
 - PickRandomPerson
 - Events
 - event
 - Environments
 - Embedded Objects
 - Analysis Data
 - Presentation
 - Person
 - Statecharts
 - Simulation: Main
 - HybridABMNetworkModeling*
 - Main
 - Ports
 - Environments
 - Embedded Objects
 - Analysis Data
 - Presentation
 - Person
 - AgentEntity
 - EntityStats
 - Simulation: Main

Main

EntityStats.java

AgentEntity.java

Person

networkResourcePool

Resource utilization: 0.569

environment

Population

TimeInSystemHist

Time in System

Properties

Console

Population - Person

General

Name: Population ☒ Show name ☐ Ignore ☐ Public ☒ Show at runtime Create presentation

Type: Person

Package: hybridabmnetworkmodeling

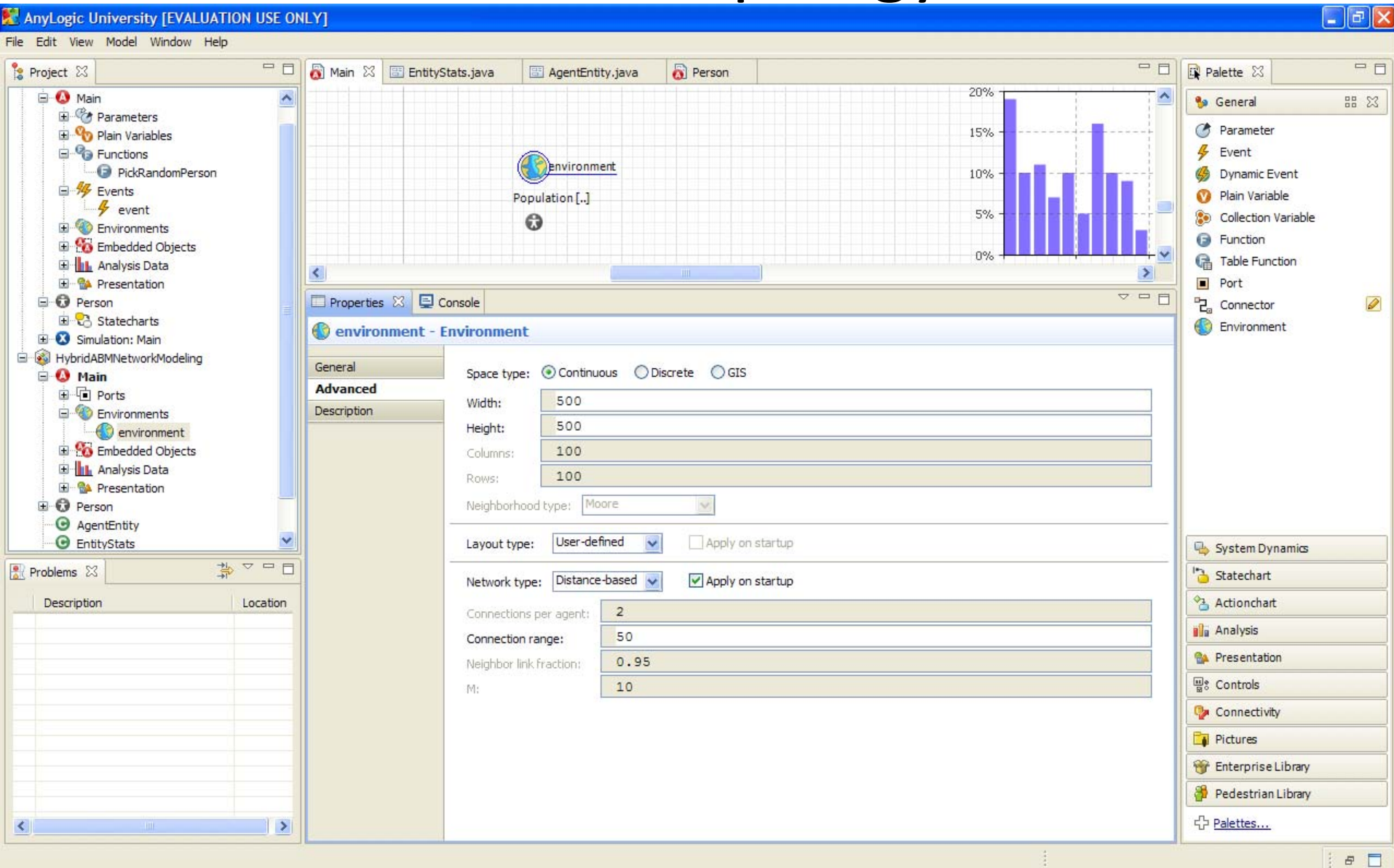
Environment: environment

Replication: 100

System Dynamics

- Statechart
- Actionchart
- Analysis
- Presentation
- Controls
- Connectivity
- Pictures
- Enterprise Library
- Pedestrian Library
- Palettes...

Based Topology



Add a “Color” variable

The screenshot displays the AnyLogic University interface with the following components:

- Project Explorer (Left):** Shows a hierarchical tree of the simulation model. The 'Person' entity is expanded, showing 'Plain Variables' and 'Statecharts'. The 'color' variable is listed under 'Plain Variables'.
- Main Canvas (Center):** Displays a statechart diagram with two states: 'Susceptible' and 'Infected'. A 'color' variable is shown as a circle with a 'V' icon on the canvas.
- Properties Panel (Bottom Center):** Shows the 'color - Plain Variable' properties. The 'Name' is 'color', 'Access' is 'public', and 'Type' is 'Color'. The 'Initial value' is set to 'BLACK'.
- Palette (Right):** Lists various simulation elements including 'Parameter', 'Event', 'Dynamic Event', 'Plain Variable', 'Collection Variable', 'Function', 'Table Function', 'Port', 'Connector', and 'Environment'.
- Problems Panel (Bottom Left):** Lists several errors: 'COLOR cannot be resolved to a ...' in 'Hybrid...'. These errors are likely due to the variable not being properly defined or linked in the statechart.

Set Dynamic Property “Color” of Oval

The screenshot displays the AnyLogic University software interface, which is used for modeling and simulation. The main workspace shows a statechart diagram with two states, "Susceptible" and "Infected", connected by transitions. A variable named "color" is defined in the "EntityStats.java" file. The "Person" entity is selected, and its properties are visible in the "Properties" pane. The "Dynamic" tab is active, showing the "Fill color" property set to "color".

AnyLogic University [EVALUATION USE ONLY]

File Edit View Model Window Help

Project

- Plain Variables
- Functions
 - PickRandomPerson
- Events
 - event
- Environments
- Embedded Objects
- Analysis Data
- Presentation
- Person
 - Statecharts
- Simulation: Main
- HybridABMNetworkModeling*
- Main
 - Ports
 - Environments
 - environment
 - Embedded Objects
 - Analysis Data
 - Presentation
- Person
 - AgentEntity
 - EntityStats
 - Simulation: Main

Main EntityStats.java AgentEntity.java Person

statechart

Susceptible

Infected

color

Properties Console

oval - Oval

General	Radius X:	
Advanced	Radius Y: <th></th>	
Dynamic	Replication: <th></th>	
Description	Visible: <th></th>	
	X: <th></th>	
	Y: <th></th>	
	Fill color:	color
	On click:	

System Dynamics

- Statechart
- Actionchart
- Analysis
- Presentation
- Controls
- Connectivity
- Pictures
- Enterprise Library
- Pedestrian Library
- Palettes...

Set "Susceptible" to Turn Color Green

AnyLogic University [EVALUATION USE ONLY]

File Edit View Model Window Help

Project

- Plain Variables
- Functions
 - PickRandomPerson
- Events
 - event
- Environments
- Embedded Objects
- Analysis Data
- Presentation
- Person
 - Statecharts
 - Simulation: Main
- HybridABMNetworkModeling*
- Main
 - Ports
 - Environments
 - environment
 - Embedded Objects
 - Analysis Data
 - Presentation
- Person
 - AgentEntity
 - EntityStats
 - Simulation: Main

Main EntityStats.java AgentEntity.java Person

Palette

- General
 - Parameter
 - Event
 - Dynamic Event
 - Plain Variable
 - Collection Variable
 - Function
 - Table Function
 - Port
 - Connector
 - Environment
- System Dynamics
 - Statechart
 - Actionchart
 - Analysis
 - Presentation
 - Controls
 - Connectivity
 - Pictures
 - Enterprise Library
 - Pedestrian Library
 - Palettes...

Susceptible - State

General

Name: Susceptible ☒ Show name ☐ Ignore ☐ Public ☒ Show at runtime

Fill color: Default

Entry action:

color=GREEN

Exit action:

Susceptible

Infected

statechart

color

Selection X=26, Y=225

Set “Infected” to Turn Color Red

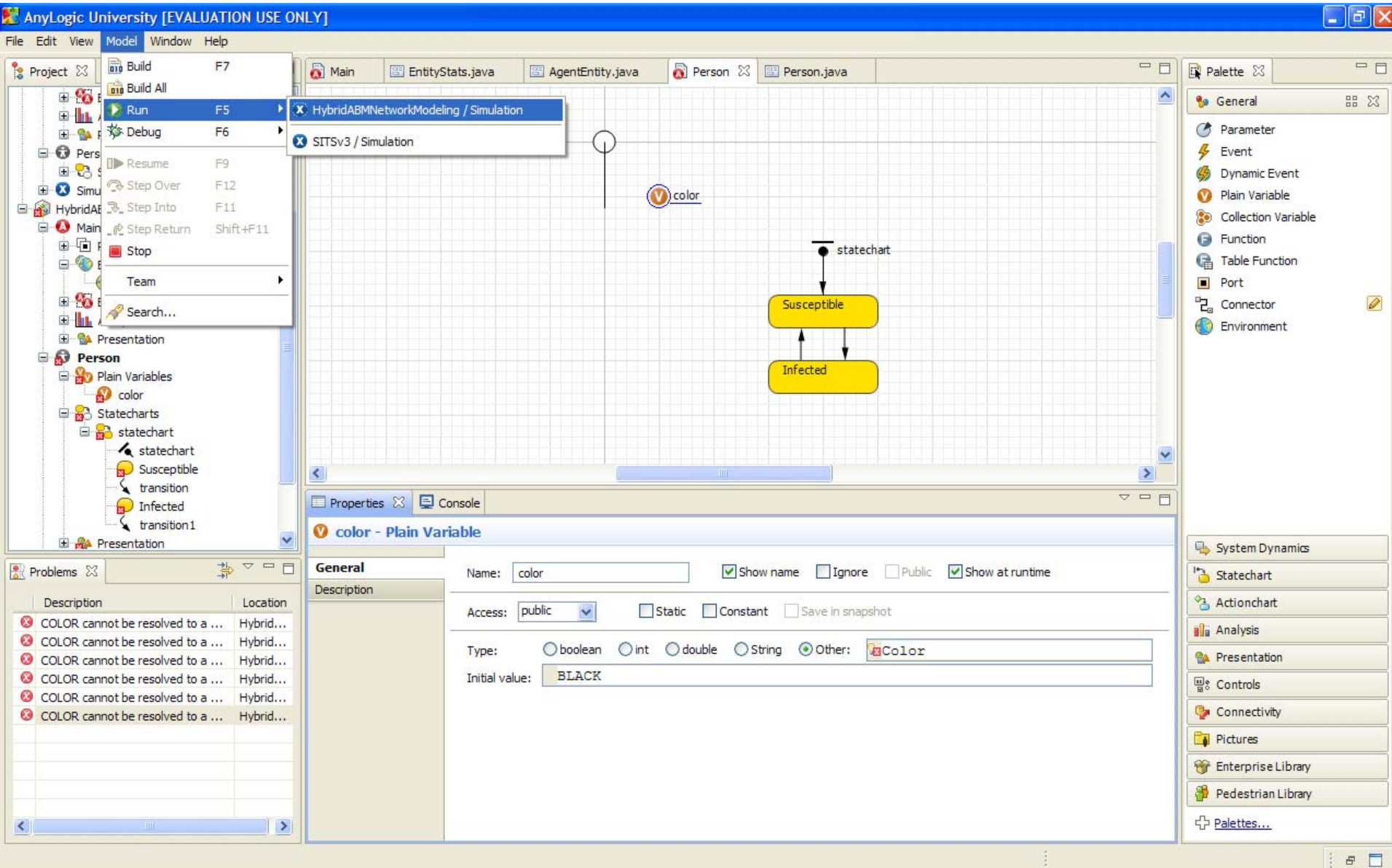
The screenshot shows the AnyLogic University software interface. The main workspace displays a statechart for a **Person** entity. The statechart has two states: **Susceptible** and **Infected**. The **Infected** state is currently selected, and its properties are shown in the bottom panel.

Infected - State Properties:

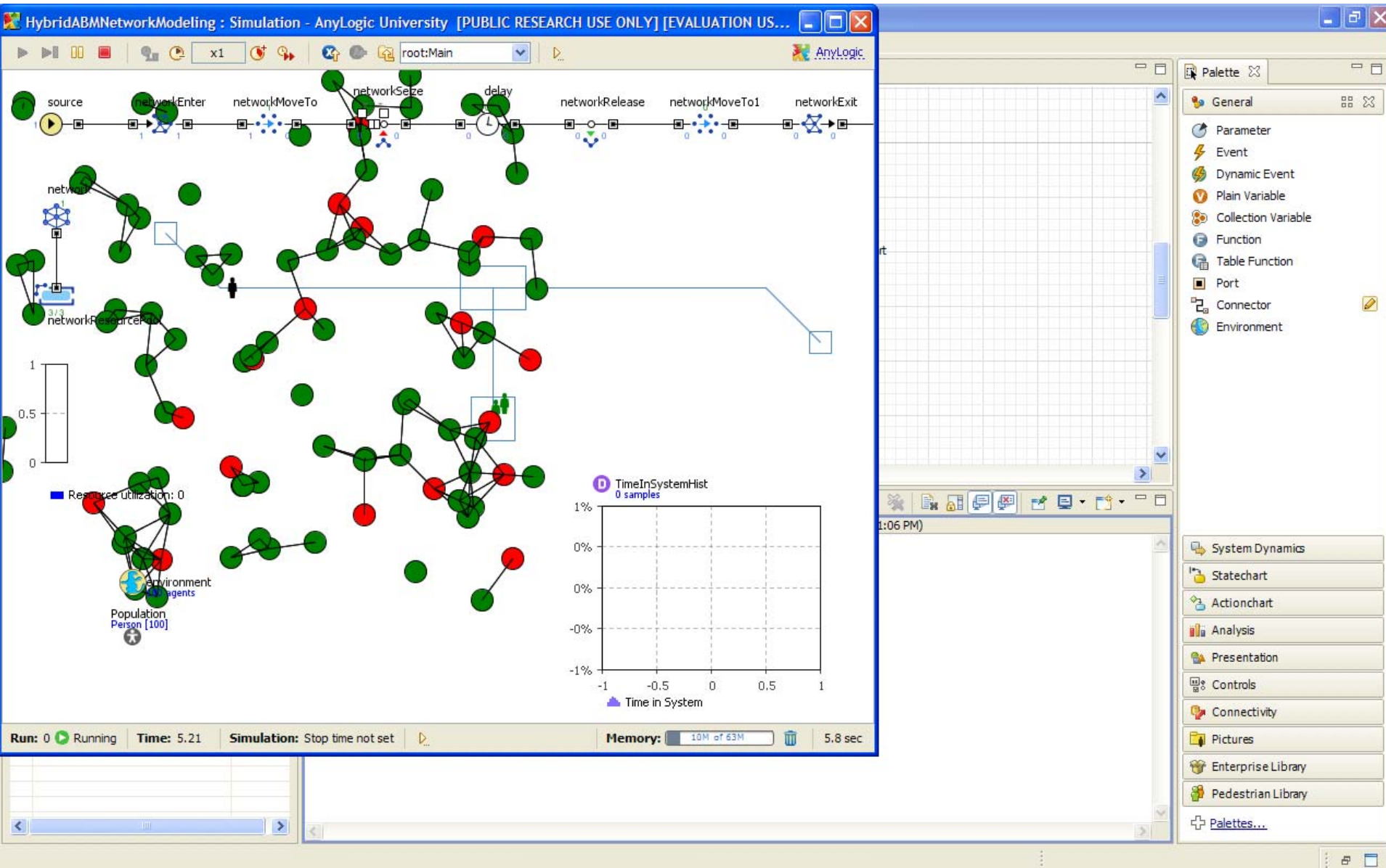
- Name: **Infected**
- Show name: ☒
- Ignore: ☐
- Public: ☐
- Show at runtime: ☒
- Fill color: **Default**
- Entry action: **color=RED**
- Exit action: (empty)

The **Project** pane on the left shows the hierarchy of the model, including **Person** and **Simulation: Main**. The **Palette** on the right shows various components like **Parameter**, **Event**, **Dynamic Event**, **Plain Variable**, **Collection Variable**, **Function**, **Table Function**, **Port**, **Connector**, and **Environment**.

Test the Current Model by Running It



Simultaneous Models – Not Yet Linked!



Set Transition to Call a Method to Inject an Agent

The screenshot displays the AnyLogic University software interface, which is used for modeling and simulation. The main workspace shows a statechart with two states: "Susceptible" and "Infected". A transition arrow points from "Susceptible" to "Infected". The transition is configured in the "transition - Transition" panel at the bottom.

transition - Transition

General

Name: ☐ Show name ☐ Ignore ☐ Public ☒ Show at runtime

Triggered by:

Rate:

Action:

Guard:

The interface also includes a Project tree on the left, a Palette on the right, and a Problems panel at the bottom left.

Set “Cure” of Infection to Depend on a Message (a Treatment Message)

The screenshot displays the AnyLogic University interface for modeling a simulation. The main workspace shows a statechart for a **Person** entity. The statechart starts at a **statechart** entry point, leading to the **Susceptible** state. A transition (labeled **transition1**) leads from **Susceptible** to the **Infected** state. The transition is triggered by a message.

The **transition1 - Transition** properties panel is visible, showing the following configuration:

- Name:** transition1
- Triggered by:** Message
- Message type:** boolean, int, double, String, Other
- Class name:** Object
- Fire transition:** Unconditionally
- Action:** `traceln("I was cured!")`
- Guard:** (empty)

The left sidebar shows the project structure, including the **Person** entity and its statechart. The right sidebar shows the palette with various modeling elements like Statechart, State, Transition, and Initial State Pointer.

Making Sure the Statechart for “Person” Gets the Message sent to a person

The screenshot displays the AnyLogic University [EVALUATION USE ONLY] interface. The main workspace shows a statechart for a 'Person' agent. The statechart has two states: 'Susceptible' and 'Infected', both represented by yellow rounded rectangles. A transition arrow points from 'Susceptible' to 'Infected'. Above the states, a 'statechart' entry point is shown with a downward arrow pointing to the 'Susceptible' state. A variable 'color' is also visible in the workspace.

The left sidebar shows the Project tree with the following structure:

- Project
 - sink
 - network
 - networkSeize
 - delay
 - networkRelease
 - networkMoveTo1
 - networkResourcePool
 - Population
 - Analysis Data
 - Presentation
 - Person**
 - Plain Variables
 - color
 - Statecharts
 - statechart
 - Susceptible
 - transition
 - Infected
 - transition1
 - Presentation
 - AgentEntity
 - EntityStats
 - Simulation: Main

The bottom panel shows the 'Person - Active Object Class' properties. The 'Agent' tab is selected, showing the following fields:

- Initial coordinates:
 - X:
 - Y:
- Movement parameters:
 - Velocity:
 - Rotation:
- On arrival:
- On message received:
 - `statechart.receiveMessage(msg);`
- On before step:
- On step:

The right sidebar shows the Palette with the following categories:

- General
- System Dynamics
- Statechart
 - Statechart Entry Point
 - State
 - Transition
 - Initial State Pointer
 - Branch
 - History State
 - Final State
- Actionchart
- Analysis
- Presentation
- Controls
- Connectivity
- Pictures
- Enterprise Library
- Pedestrian Library
- Palettes...

The bottom status bar shows 'Person - ActiveObjectClass'.

Double Click on Main Object & Select “Source” & Set to Manual Injection

The screenshot displays the AnyLogic University [EVALUATION USE ONLY] interface. The main workspace shows a simulation model with a central 'source' object (a yellow circle with a play button) connected to a sequence of network-related objects: 'networkEnter', 'networkMoveTo', 'networkSeize', 'delay', 'networkRelease', and 'networkMoveTo'. A 'networkResourcePool' is also visible. The left sidebar contains a project tree with folders like 'Main', 'Person', 'AgentEntity', and 'EntityStats'. The bottom panel shows the 'source - Source' properties window.

source - Source Properties

- General**
 - ☒ Show name ☐ Ignore ☐ Public ☒ Show at runtime [Create presentation](#)
- Parameters**
 - Entity class: `Entity`
- Statistics**
 - Entity class: `Entity`
- Description**
 - Entity class: `Entity`
- Injection**
 - Injection type: ☐ Rate ☐ Interarrival time ☐ Rate table ☐ Arrival table ☒ Manual (call inject() method)
- Code**
 - new EntityStats(time())
- Diagram**
 - Diagram shape: `groupEntity`

In “Main”, Create a Variable To Inject in the Movement Network

The screenshot displays the AnyLogic University interface with the 'Main' model open. The central workspace shows a network diagram with a 'source' node, a 'networkEnter' node, a 'networkMoveTo' node, a 'networkSeize' node, a 'delay' node, a 'networkRelease' node, and a 'networkMoveTo' node. A 'networkResourcePool' is also visible. The 'agentToInjectTransient' variable is highlighted in the diagram.

The left sidebar shows the project structure, including 'Main' and 'AgentEntity'. The 'Main' model contains 'Plain Variables', 'Ports', 'Functions', 'Environments', 'Embedded Objects', 'Analysis Data', and 'Presentation'. The 'AgentEntity' model contains 'Plain Variables', 'Statecharts', 'statechart', 'Susceptible', 'transition', 'Infected', and 'transition1'.

The bottom panel shows the 'Properties' window for the 'agentToInjectTransient' variable. The 'General' tab is selected, showing the following properties:

- Name: agentToInjectTransient
- Show name: ☒ Show name
- Ignore: ☐ Ignore
- Public: ☐ Public
- Show at runtime: ☒ Show at runtime
- Access: public
- Static: ☐ Static
- Constant: ☐ Constant
- Save in snapshot: ☐ Save in snapshot
- Type: boolean, int, double, String, Other: **Person**
- Initial value: null

The bottom status bar shows 'agentToInjectTransient - Plain Variable' and 'Selection'.

Create Function in “Main”: Part 1

AnyLogic University [EVALUATION USE ONLY]

File Edit View Model Window Help

Project

- Person
 - Statecharts
 - Simulation: Main
 - HybridABMNetworkModeling*
 - Main
 - Ports
 - Functions
 - Environments
 - environment
 - Embedded Objects
 - Analysis Data
 - Presentation
 - Person
 - Plain Variables
 - color
 - Statecharts
 - statechart
 - statechart
 - Susceptible
 - transition
 - Infected
 - transition1
 - Presentation
 - AgentEntity
 - EntityStats

Main

EntityStats.java

AgentEntity.java

Person

InjectAgentEntity

source

networkEnter

networkMoveTo

networkSeize

delay

networkRelease

networkMoveTo

network

networkResourcePool

Properties

Console

InjectAgentEntity - Function

General

Name: InjectAgentEntity

☒ Show name ☐ Ignore ☐ Public ☒ Show at runtime

Access: default ☐ Static

Return type: ☒ void ☐ boolean ☐ int ☐ double ☐ String ☐ Other: void

Function arguments:

Name	Type
p	Person

System Dynamics

- Statechart
- Actionchart
- Analysis
- Presentation
- Controls
- Connectivity
- Pictures
- Enterprise Library
- Pedestrian Library
- Palettes...

Problems

Description	Location
Parameter type is not specified.	Hybrid...
Type mismatch: cannot convert f...	Hybrid...
Syntax error on token ":", ++ e...	Hybrid...

Create Function in Main Part 2

The screenshot displays the AnyLogic University [EVALUATION USE ONLY] interface. The main workspace shows a simulation model with a flowchart for agent injection. The flowchart starts with a 'source' block, followed by a 'networkEnter' block, then a 'networkMoveTo' block, a 'networkSeize' block, a 'delay' block, a 'networkRelease' block, and finally a 'networkMoveTo' block. A 'networkResourcePool' block is also visible. The 'InjectAgentEntity' function is highlighted in the flowchart. The 'Properties' panel shows the 'InjectAgentEntity' function with the following code:

```
Function body:  
agentToInjectTransient = p;  
source.inject(1);
```

The 'Project' panel on the left shows the hierarchy of the simulation model, including 'Main', 'HybridABMNetworkModeling*', 'Person', 'Plain Variables', 'Ports', 'Functions', 'Environments', 'environment', 'Embedded Objects', 'Analysis Data', 'Presentation', 'Person', 'Plain Variables', 'color', 'Statecharts', 'statechart', 'statechart', 'Susceptible', 'transition', 'Infected', 'transition1', 'Presentation', and 'AgentEntity'. The 'Problems' panel at the bottom left shows two errors: 'Type mismatch: cannot convert f...' and 'Syntax error on token *, ++e...'. The 'Palette' panel on the right shows various blocks and connectors available for use.

Set “Source” to Inject an Entity Associated with the Desired Agent

The screenshot displays the AnyLogic University interface, showing a simulation model setup for injecting an entity into a network. The main workspace contains a flowchart with the following components and connections:

- InjectAgentEntity** (Function block) is connected to **networkEnter** (Network block).
- networkEnter** is connected to **networkMoveTo** (Network block).
- networkMoveTo** is connected to **networkSeize** (Network block).
- networkSeize** is connected to **delay** (Delay block).
- delay** is connected to **networkRel** (Network block).
- networkRel** is connected to **networkResourcePool** (Network block).
- networkResourcePool** is connected to **network** (Network block).
- network** is connected to **networkSeize**.

The **source** block is highlighted, and its properties are shown in the **Properties** panel:

source - Source

- Name:** source
- Type:** Source<T extends Entity>
- Entity class:** Entity
- Package:** com.xj.anylogic.libraries.enterprise
- Arrivals defined by:** Manual (call)
- Limited number of arrivals:** ☐
- New entity:** `new AgentEntity(agentToInjectTransient)`
- On exit:**
- Entity animation shape:** groupEntity

The **Palette** on the right shows various blocks available for use, including Parameter, Event, Dynamic Event, Plain Variable, Collection Variable, Function, Table Function, Port, Connector, and Environment.

The **Project** pane on the left shows the hierarchy of the simulation model, including Main, Simulation: Main, HybridABMNetworkModeling*, and Person.

The **Problems** pane at the bottom left shows two errors:

- Type mismatch: cannot convert f... Hybrid...
- Syntax error on token ":", ++ e... Hybrid...

Set Agents to be Cured After Having been Seen by Doctors!

AnyLogic University [EVALUATION USE ONLY]

File Edit View Model Window Help

Project HybridABMNetworkModeling*

- Main
 - Plain Variables
 - Ports
 - Functions
 - Environments
 - environment
 - Embedded Objects
 - source
 - networkEnter
 - networkMoveTo
 - networkExit
 - sink
 - network
 - networkSeize
 - delay
 - networkRelease
 - networkMoveTo1
 - networkResourcePool
 - Population
 - Analysis Data
 - Presentation
 - Person
 - Plain Variables
 - color

Problems

Main EntityStats.java AgentEntity.java Person

Palette

- General
 - Parameter
 - Event
 - Dynamic Event
 - Plain Variable
 - Collection Variable
 - Function
 - Table Function
 - Port
 - Connector
 - Environment
- System Dynamics
- Statechart
- Actionchart
- Analysis
- Presentation
- Controls
- Connectivity
- Pictures
- Enterprise Library
- Pedestrian Library
- Palettes...

delay - Delay

General

Name: delay ☒ Show name ☐ Ignore ☐ Public ☒ Show at runtime Create present...

Parameters

Statistics

Description

Type: Delay<T extends Entity> Entity class: Entity

Package: com.xj.anylogic.libraries.enterprise

Delay time is ☒ Specified explicitly ☐ Path length / speed

Delay time^D triangular(1, 2, 3)

Maximum capacity ☒

On enter^D

On exit^D ((AgentEntity) entity).Cure()

Diagram description: The diagram shows a flow starting from a source, passing through networkEnter, networkMoveTo, networkSeize, a delay block, and networkRel. The delay block is highlighted with a yellow circle and a red box. The flow ends at a sink. A networkResourcePool is also shown.

Running the Simulation

The screenshot displays the AnyLogic University interface, showing a simulation of a network model. The main window is titled "HybridABMNetworkModeling : Simulation - AnyLogic University [PUBLIC RESEARCH USE ONLY] [EVALUATION US...".

Left Panel (Project Explorer): Shows the project structure. The "Person" entity is expanded, showing "Plain Variables" (color, statechart) and "Statecharts" (statechart, Susceptible, transition, Infected, transition1). The "Simulation: Main" entity is selected.

Center Panel (Simulation View): Displays the network model. It features a central "network" entity with a "networkResourcePool" and a "networkExit" node. The network is composed of red and green nodes, representing different states of the population. A "Population Person [100]" is shown at the bottom. The simulation is running, with a "Time: 26.00" and "Simulation: Stop time not set" status.

Right Panel (Console): Shows the simulation log. It contains messages such as "Sent cure message to pers", "I was cured!", and "Resource utilization: 0.346".

Bottom Panel (Status Bar): Displays the simulation status. It includes a "Run: 0" button, a "Time: 26.00" indicator, a "Simulation: Stop time not set" status, a "Memory: 11M of 63M" indicator, and a "32.0 sec" timer.

Network Diagram Details: The network diagram shows a complex structure of nodes and edges. Red nodes represent one state (likely Susceptible or Infected), and green nodes represent another state (likely Cured). The nodes are connected by edges, forming a network. A "networkResourcePool" is shown, and a "networkExit" node is at the end of the network. The simulation is running, and the network is evolving over time.