INF

FAKULTÄT FÜR
INFORMATIK

AnyLogic Process Modeling Library/ Influence of Random Variables

The Animated Medical Practice

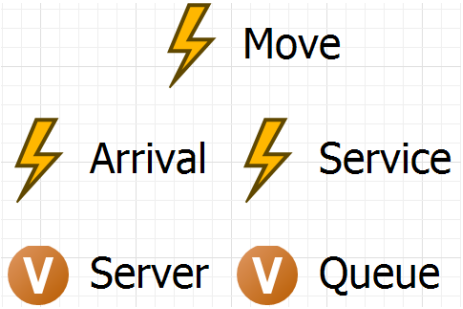
Discrete Event Modeling in AnyLogic

So far, we have modeled and analyzed system behavior using

- Variables
- Custom events

It was difficult to

- Visualize entity flow, paths, dependencies
- Discriminate or track entities



The process modeling library enables more intuitive modeling of entity–centric discrete models.

The Process Modeling Library

Models processes (e.g. manufacturing, waiting queues, …)

Fundamental objects: „Agents“

- Are persistent objects that move through a system
- Can have attributes (color, size, number of children, …)
- Can chose a path based on system state and/or their attributes
- Cannot appear or disappear at will

Limited to “material flow” systems, but easier to develop and to understand

N.B.: Some commercial tools rely on this paradigm exclusively.

Process Modeling Library Components

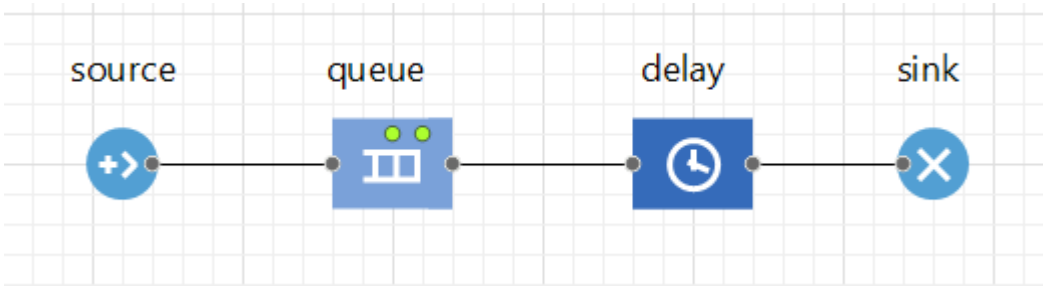
Relevant Model Components

	Source	Creates agents
	Sink	Destroys agents
	Delay	Delays agents (Activities like services etc.)
	Queue	Temporarily stores a limited number of agents
	Select Output	Lets agents choose one of two possible exits
	Hold	Can temporarily block flow of agents
	*Split*	*Forks two output entities from every input agent*
	*Combine*	*Joins two input entities to a single output agent*

All Components

- Have so–called “Ports” at which Agents enter or leave
- Ports can be connected graphically, without writing code

Bank Model using Process Modeling Library Components



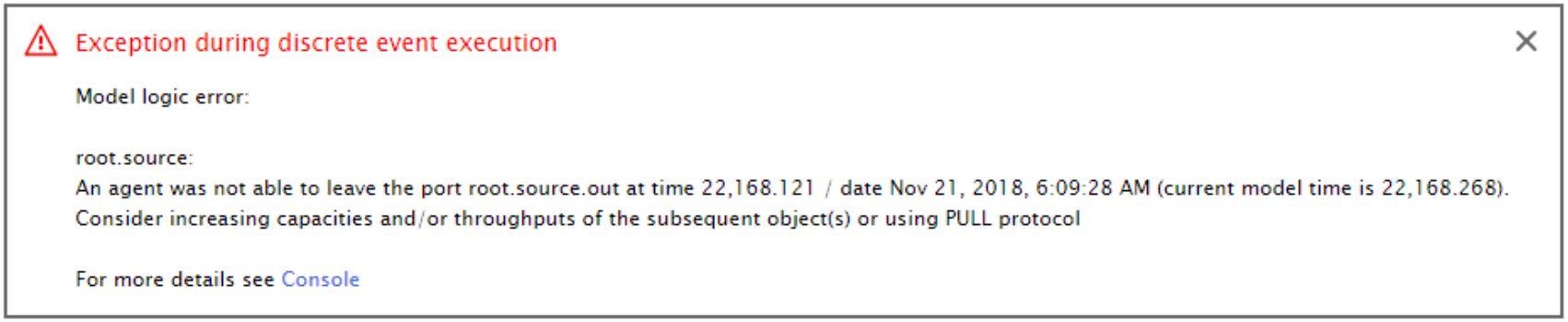
Source	Customer arrives
Queue	Customer waits
Delay	Customer is being served
Sink	Customer leaves

Result: We built a bank model

- That naturally visualizes the entity flow
- Without writing any code

Process Modeling Library Pitfalls

An agent has to always go somewhere!



Simulation crashes, if there is

- not enough space at target (e.g. queue is full)
- no component connected to an output port

The Medical Practice

The Medical Practice – Workflow

Patients:

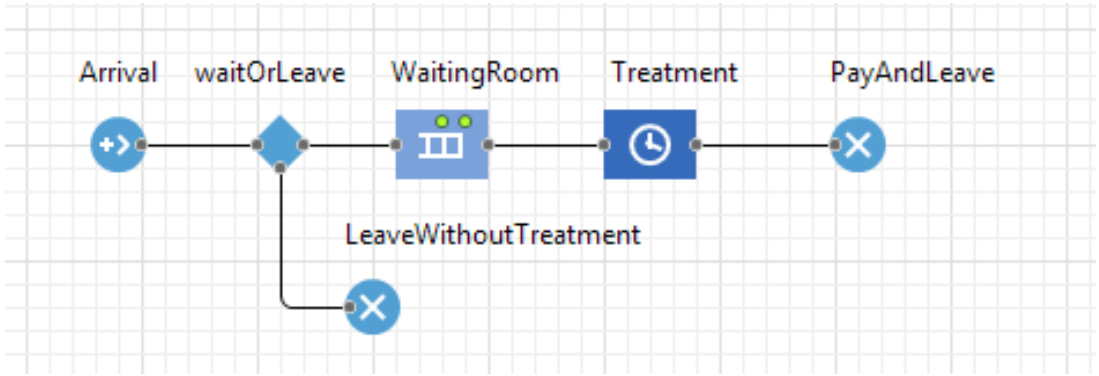
- Arrive at the practice
- If there is a seat left,
  - They wait in the waiting room
  - Otherwise, they leave at once
- They are treated by the doctor
- They pay and leave

The Medical Practice – "Material Flow"

Patients:

- Arrive at the practice
- If there is a seat left,
  - They wait in the waiting room
  - Otherwise, they leave at once
- They are treated by the doctor
- They pay and leave

The Medical Practice



Component properties to set:

Component	Property	Value (Example)
Arrival	Interarrival Time	5 minutes
WaitOrLeave	True Output Condition	WaitingRoom.size( ) < WaitingRoom.capacity
WaitingRoom	Capacity	10
Treatment	Delay Time	5 minutes

The Medical Practice

Some questions this model can help answer:

- What fraction of his work day is the doctor actually treating patients?
- How many seats in the waiting room are being used on average?
- How do these properties change when varying the arrival and treatment delay distributions?

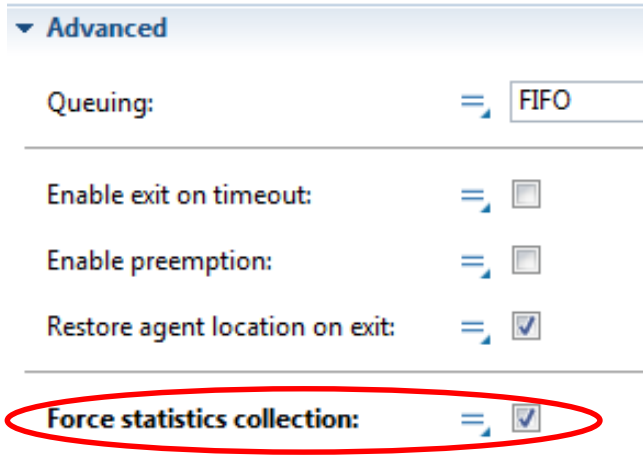
Collecting Statistics

Collecting statistics (min, max, mean, median, …) is tedious

Process Modeling library components can automatically compute these statistics

- Activated with “Force statistics collection”
- Accessed through component properties:

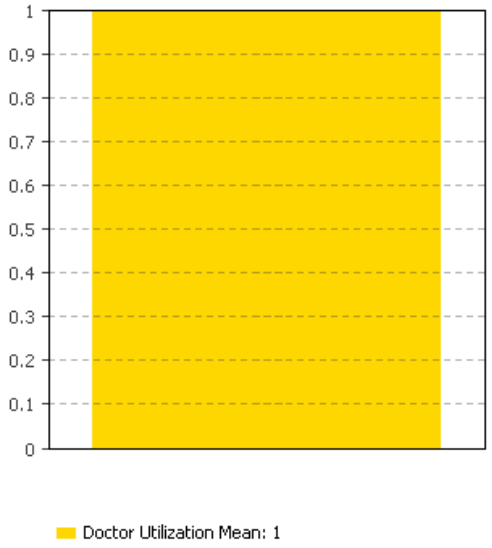
Component	Property
Delay	statsUtilization
Queue	statsSize

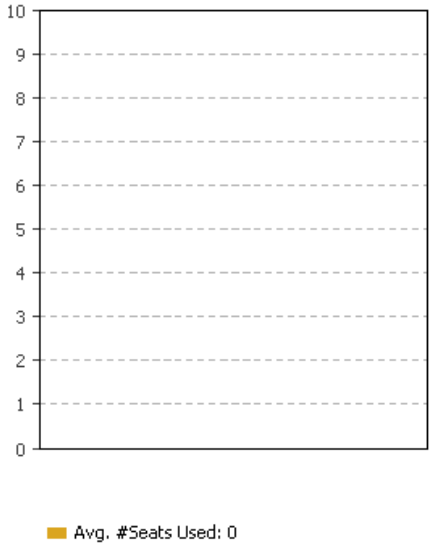


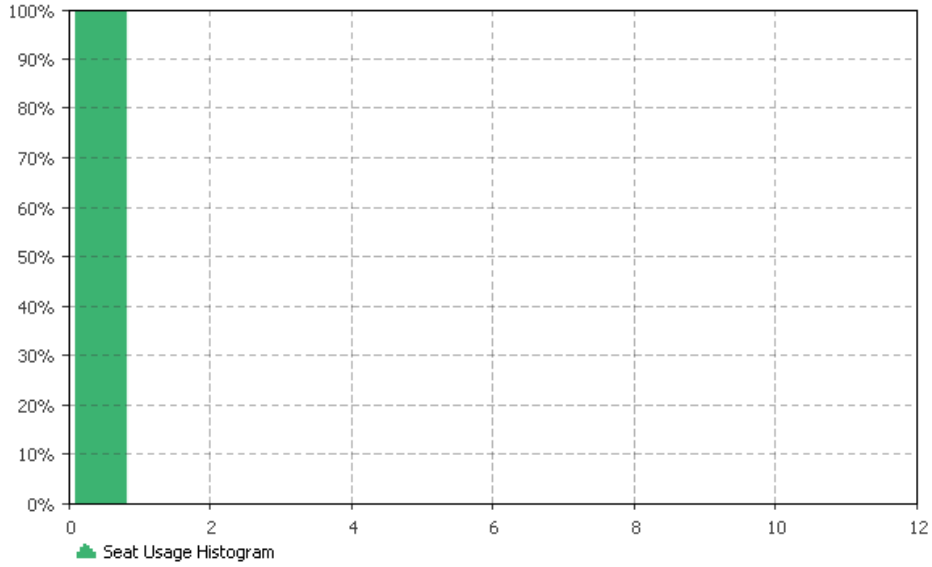
We need this at least for the Waiting Room and the Treatment stages

Some Results

#Seats:	10
Interarrival Time:	5
Delay Time:	5
Simulation End at:	100000 minutes

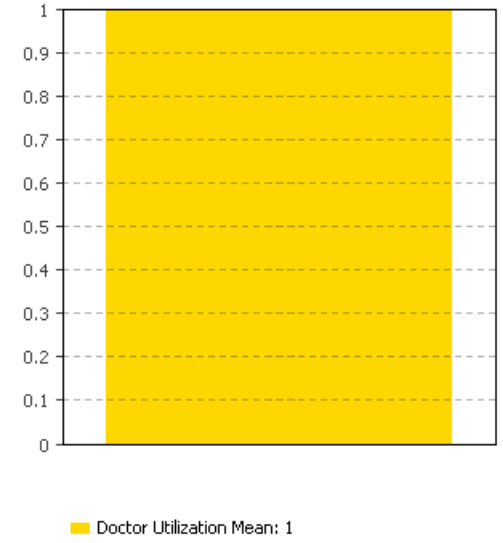


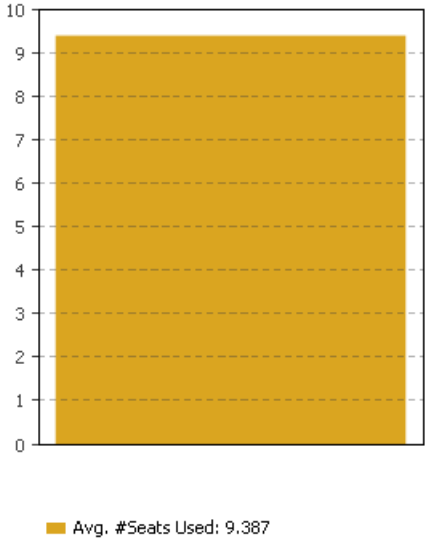


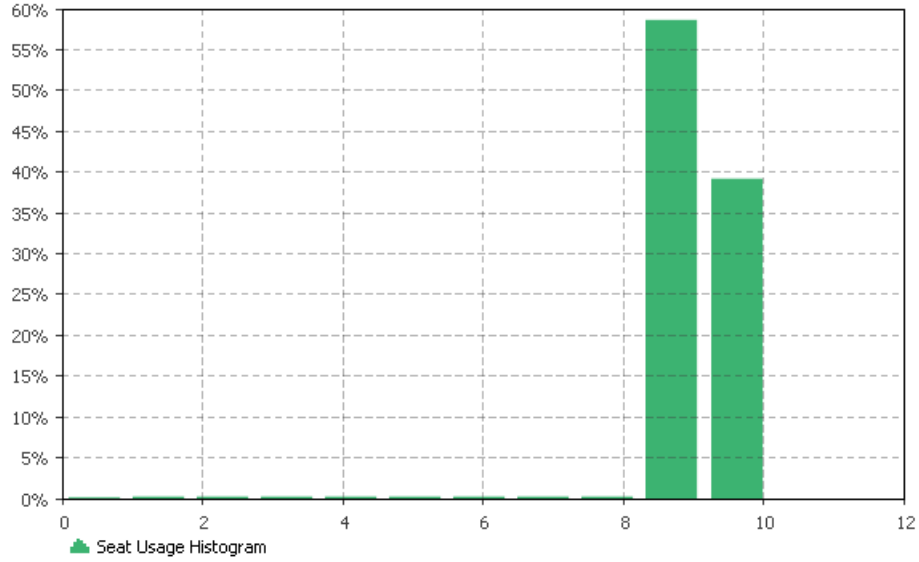


Some Results

#Seats:	10
Interarrival Time:	5
Delay Time:	5.01
Simulation End at:	100000 minutes

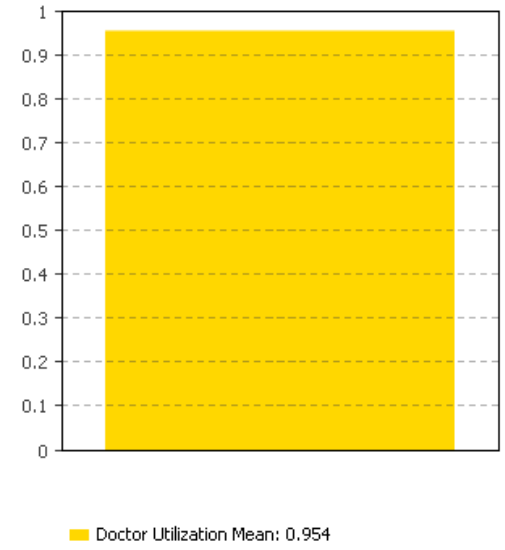


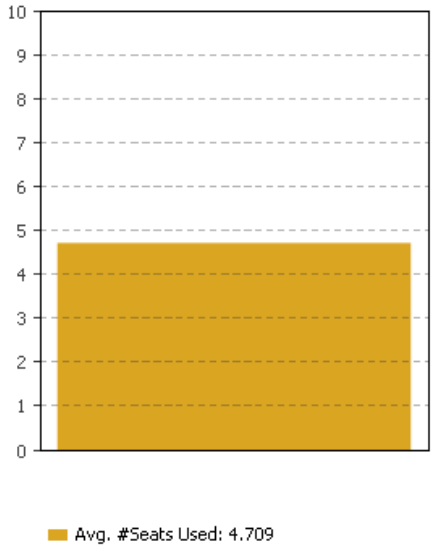


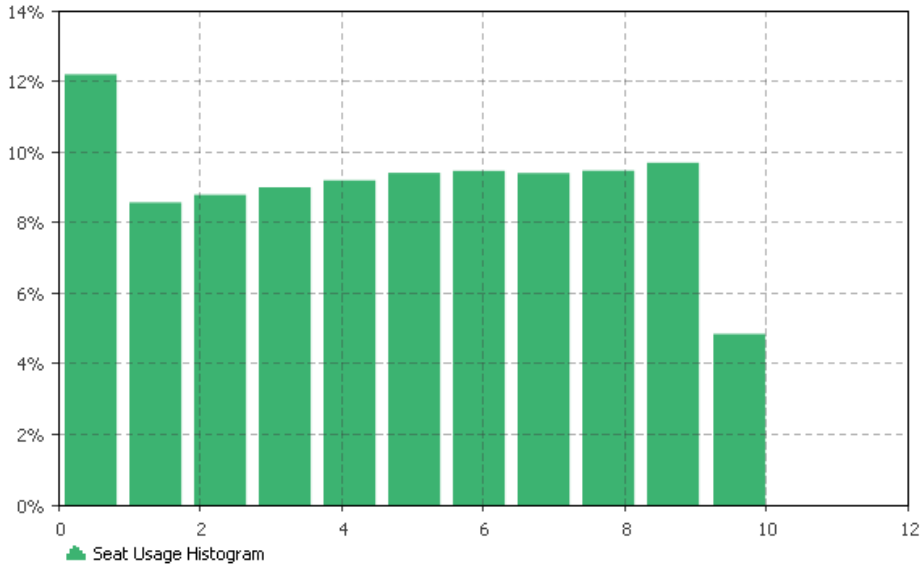


Some Results

#Seats:	10
Interarrival Time:	exponential(1/5.0)
Delay Time:	5.01
Simulation End at:	100000 minutes

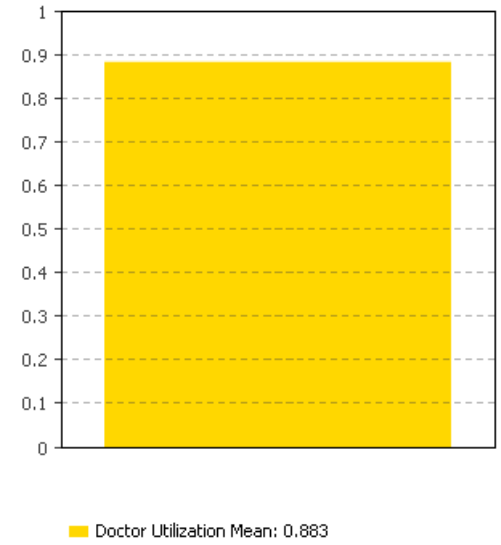


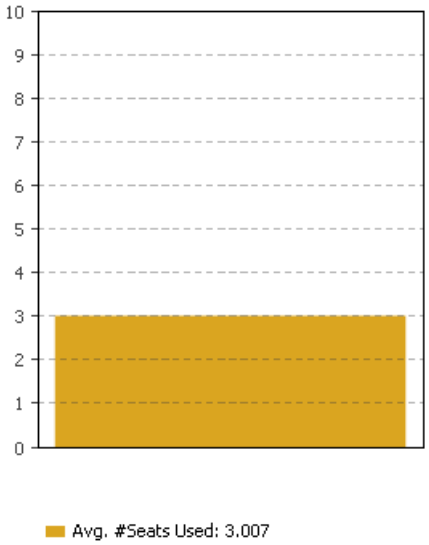


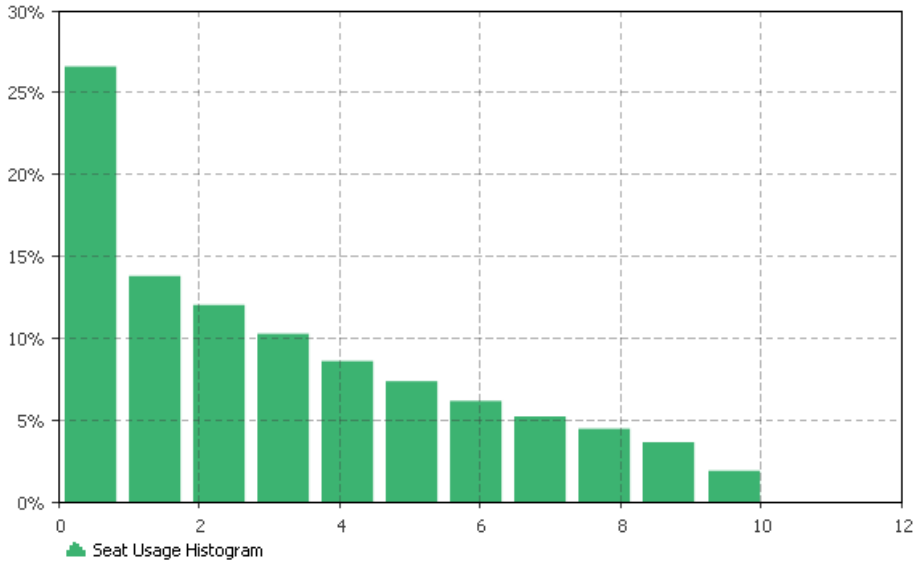


Some Results

#Seats:	10
Interarrival Time:	exponential(1/5.0)
Delay Time:	max(normal(2.0, 4.5), 0)
Simulation End at:	100000 minutes







Animations

Animations

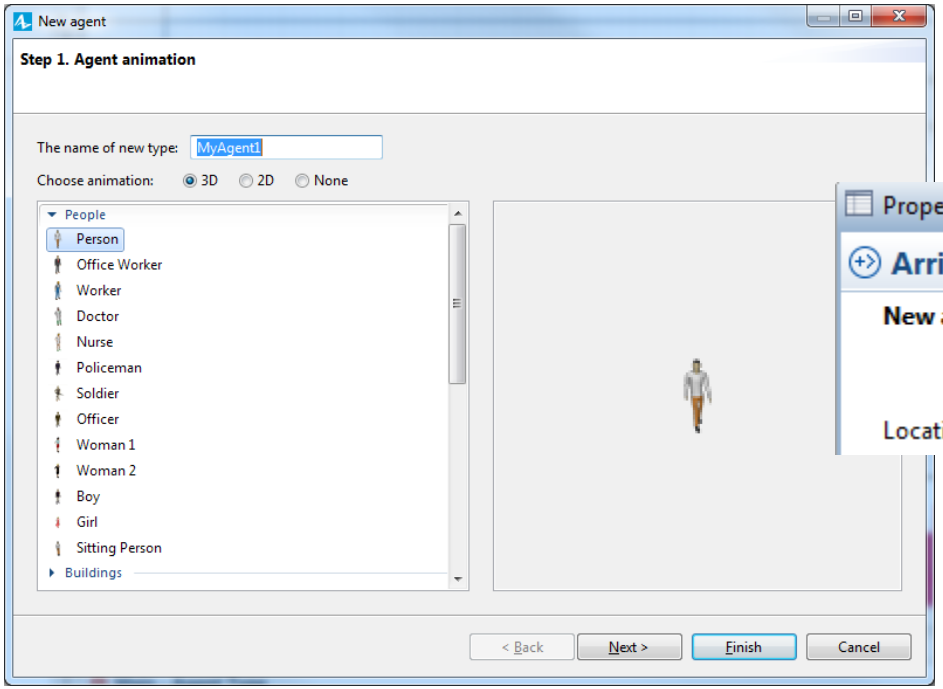
Animations can help with

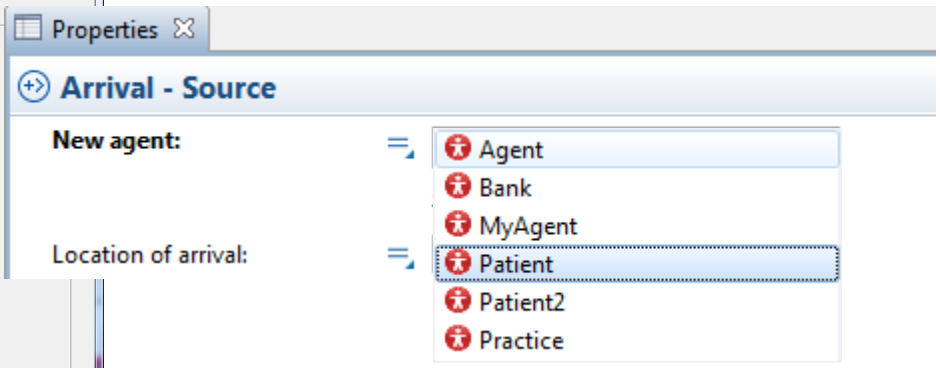
- Tracking individual entities as they move through the model
- Explaining the model to others
- Debugging and testing the model

Animations

Animating a Process Library simulation run is simple

- Create a new agent type and assign a specific shape (e.g. a human–like shape for patients)
- Select the custom agent type in the source object

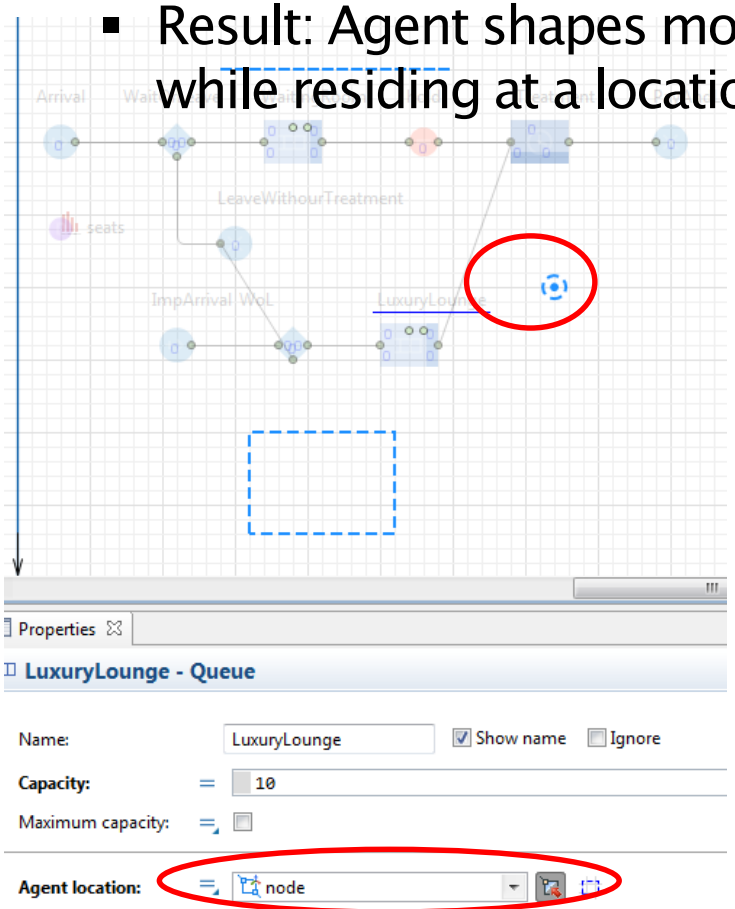


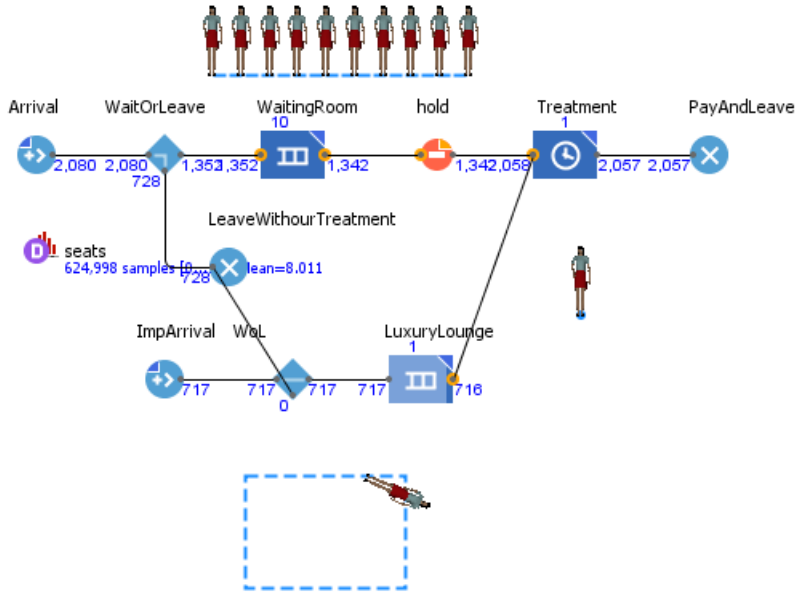


Animations

Animating a Process Library simulation run is simple

- For most locations (queues, delays) a space markup object (e.g. a simple line) can be defined
- Result: Agent shapes move along (are located in) markup object while residing at a location





A More Complicated Medical Practice – Agents with Custom Attributes

Some odd assumptions…

A two-class medical system:

- There are two types of patients: normal and important ones
- Important patients have a separate waiting room
- The doctor will not treat normal patients as long as important ones are waiting
- Treatment of important patients needs more time

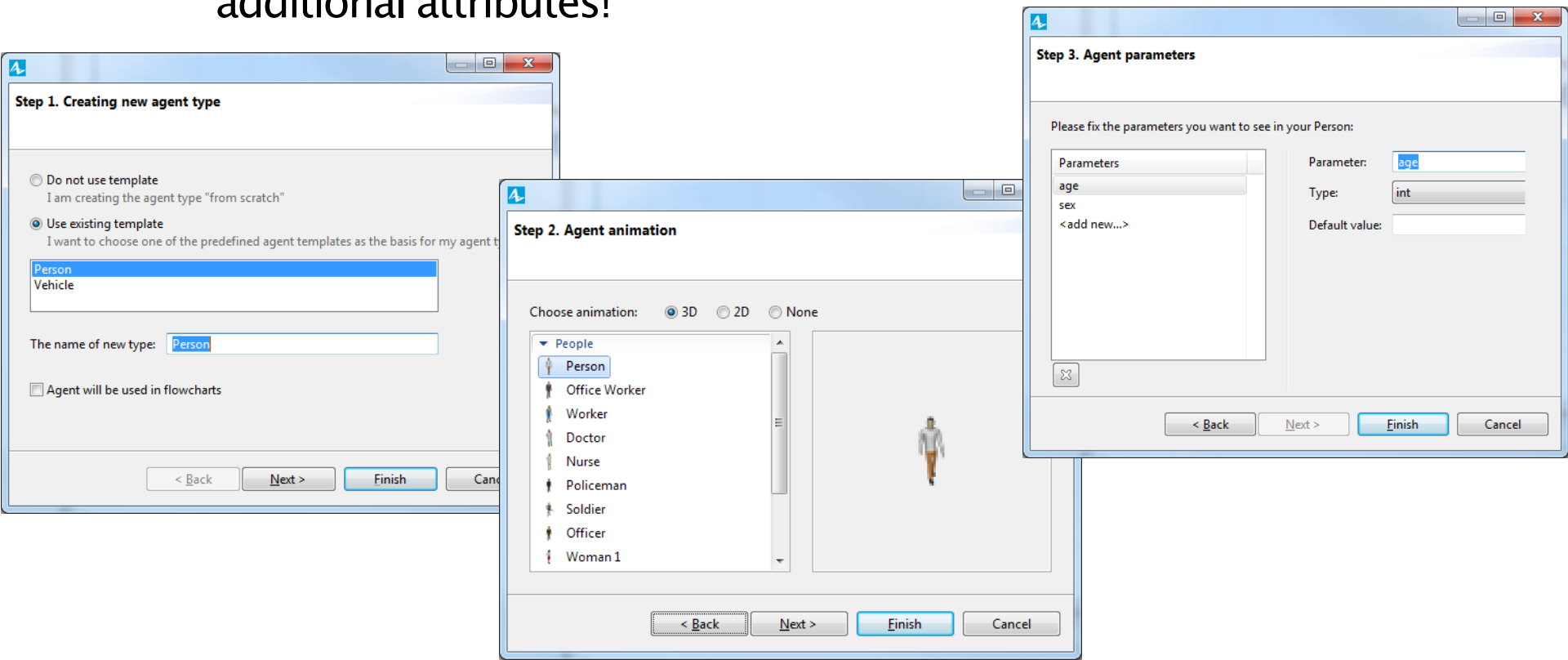
Component	Property	Value
ImpArrival	Interarrival Time	Exponential(1/15)
ImpTreatment	Delay Time	Normal(6,2)
LuxuryLounge	Capacity	10

Remembering a Patient's Importance

How can the simulation remember whether the patient to be treated is an important one or not?

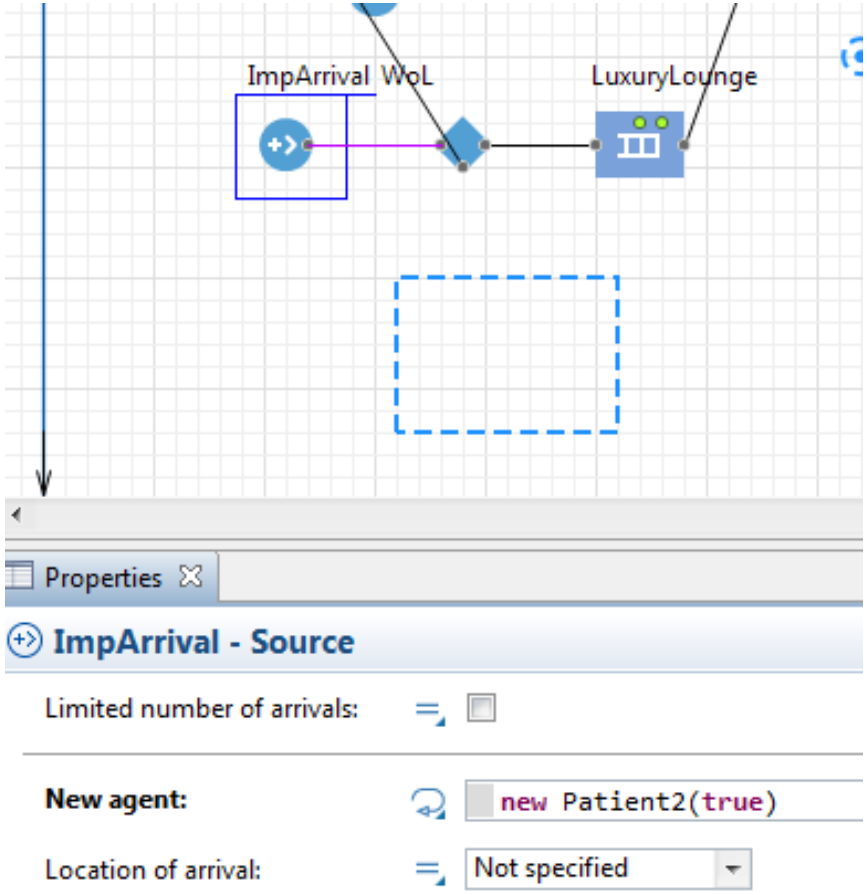
Solution:

- Remember (from the lecture): Entities can have any number of additional attributes!



Remembering a Patient's Importance

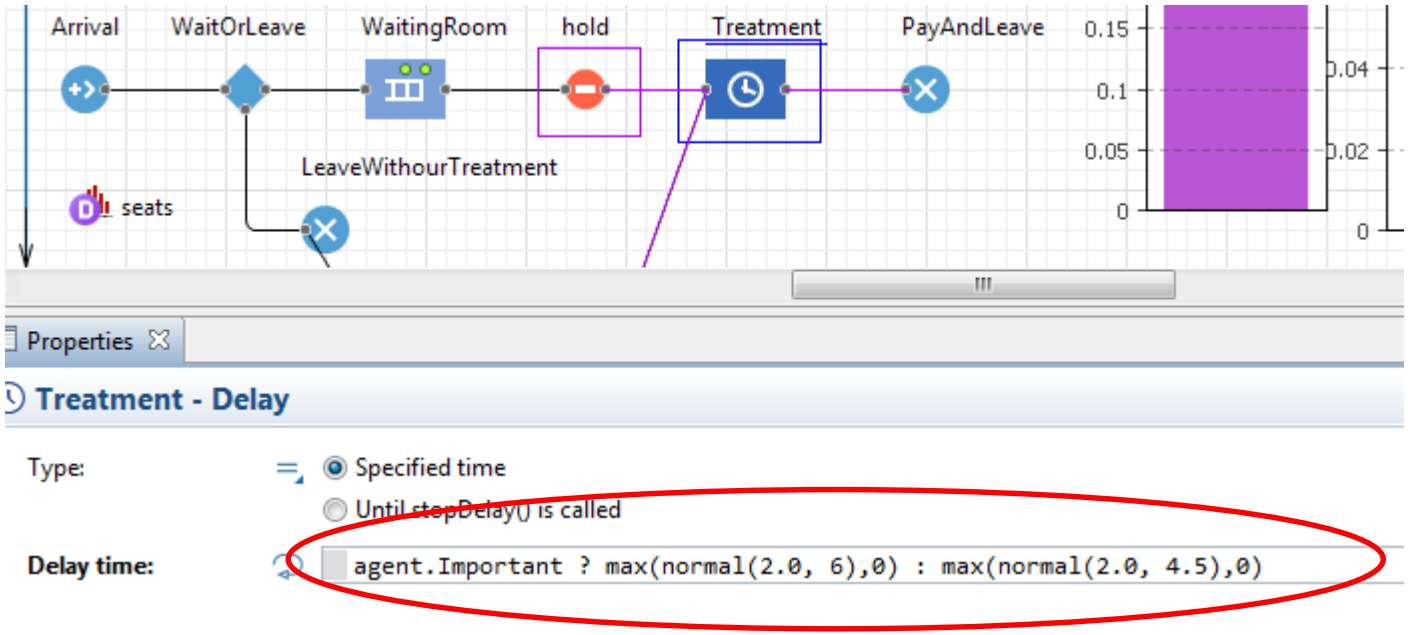
The sources make their patients store the importance



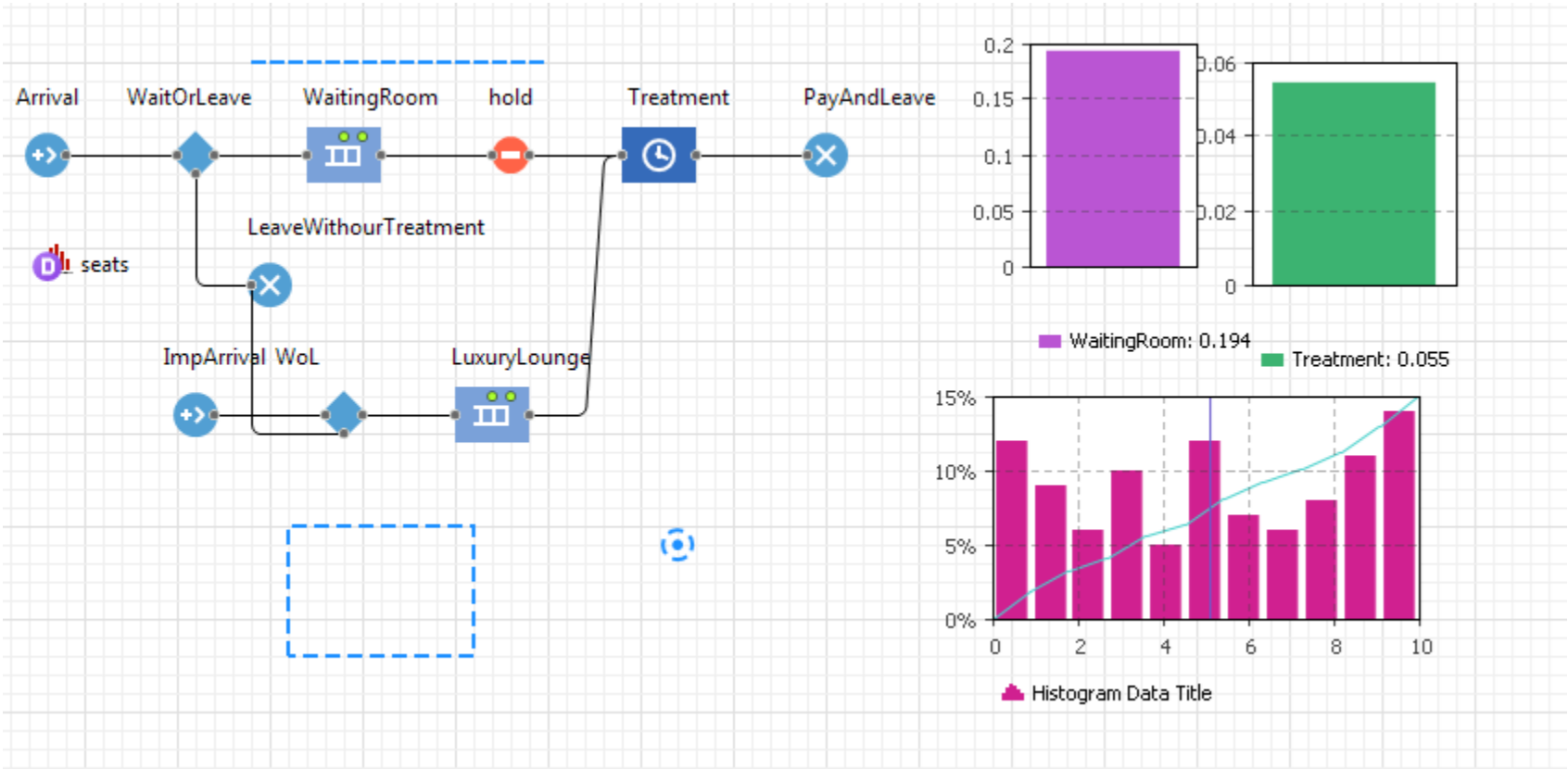
Remembering a Patient's Importance

Many components can access (and change!) the properties of entities passing through them

- If the agent type is known
- Through access to the variable "agent"



The Final Model



Simulation Results