



STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Discrete Event Simulation Using AnyLogic

*SYS-611: Simulation and
Modeling*

Paul T. Grogan, Ph.D.

Assistant Professor

School of Systems and Enterprises





Agenda

1. Introduction to AnyLogic
2. Queuing System Model
3. Inventory System Model
4. Factory System Model

Reading: [*AnyLogic in Three Days: Modeling and Simulation Textbook*](#)



Introduction to AnyLogic





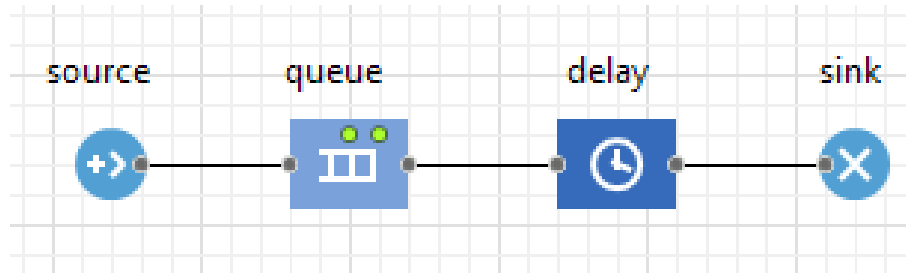
AnyLogic Overview

- **AnyLogic** is a multi-method simulation platform for models using any of the following formalisms:
 - Discrete Event Simulation
 - Agent-based Simulation
 - System Dynamics
- Uses Java under the hood and for customization
- Commercially licensed (some limitations apply)
- Free "Personal Learning Edition" available online:
 - <https://www.anylogic.com/downloads/personal-learning-edition-download/>

AnyLogic Process Modeler

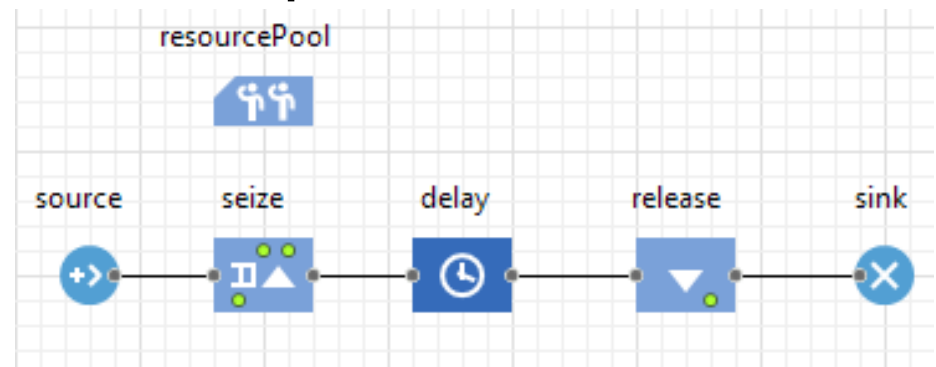
- Click-and-drag common process components:

- Sources
- Queues
- Delays
- Sinks



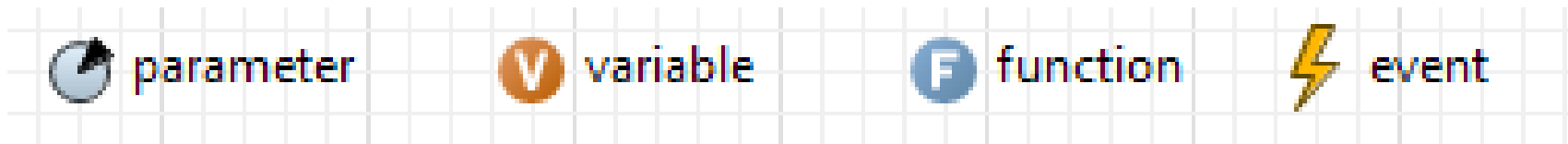
- And more complex process components:

- Resource pools
- Seizing resources
- Releasing resources



AnyLogic Agent Modeler

- Click-and-drag common “agent” components:
 - **Parameters:** scalar state variables (typically inputs)
 - **Variables:** more flexible state variable (typically outputs)
 - **Functions:** state transitions / updates
 - Note: typically have to program in Java language
 - **Events:** when to trigger functions
 - Automatic on constant timeout schedule or manually scheduled



-



Simulation Experiment

- Control simulation settings
- Model time
 - Virtual or scaled real time
 - Terminal condition
- Randomness
 - Random seed
 - Fixed seed

The screenshot shows the 'Simulation - Simulation Experiment' properties window. It contains several sections for configuring the simulation:

- Name:** Simulation (with an 'Ignore' checkbox).
- Top-level agent:** Main (dropdown menu).
- Maximum available memory:** 512 (dropdown menu) Mb.
- Model time:**
 - Execution mode:** ☒ Real time with scale 50 (dropdown menu), ☐ Virtual time (as fast as possible).
 - Stop:** Stop at specified time (dropdown menu).
 - Start time:** 0 (text input), **Stop time:** 100 (text input).
 - Start date:** 11/14/2018 (calendar icon), **Stop date:** 12/14/2018 (calendar icon).
 - Time of day:** 12:00:00 AM (time picker), **Time of day:** 12:00:00 AM (time picker).
- Randomness:**
 - Random number generation:** ☒ Random seed (unique simulation runs), ☐ Fixed seed (reproducible simulation runs) (Seed value: 1), ☐ Custom generator (subclass of Random): new Random().
 - Selection mode for simultaneous events:** LIFO (in the reverse order of scheduling).
- Window:** Java actions, Advanced Java, Advanced, Description.



AnyLogic Caveats

- Very comprehensive/complex software package
- Java programming is required for custom models
- Personal Learning Edition (PLE) is limited
 - No Monte Carlo / optimization / sensitivity analysis
 - Limitations on size/scope of simulation models
- Commercially licensed, proprietary
 - All modeling must be done within AnyLogic environment
 - Cannot export models without special license (\$\$\$)

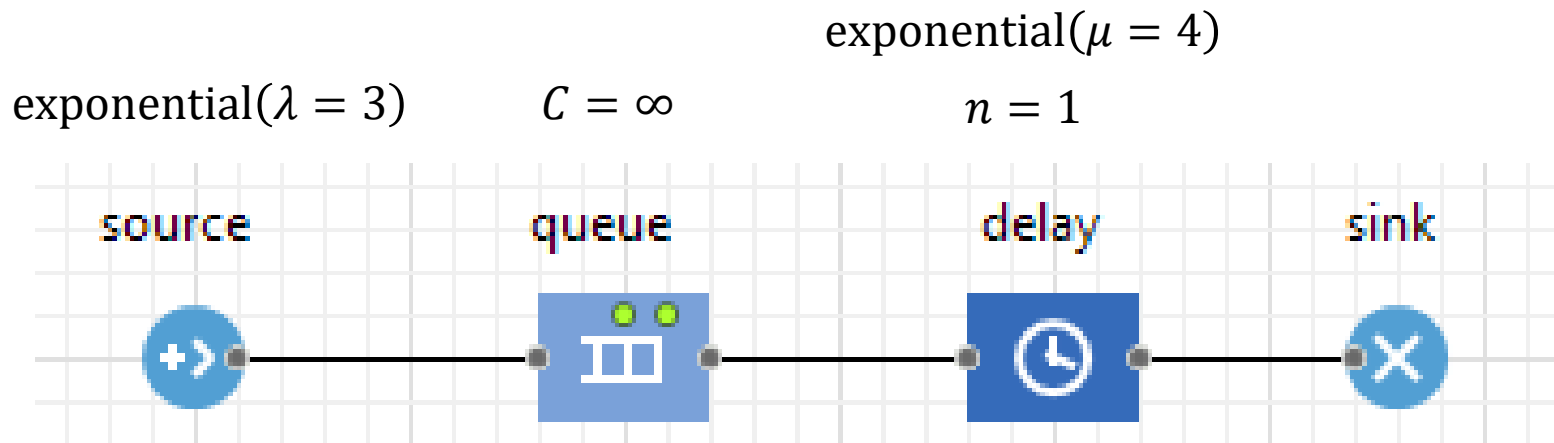


AnyLogic Example: Queuing Model



Queuing System "Process"

- Queuing system specification:
 - $n = 1$ server, $C = \infty$ queue capacity
 - Customer inter-arrival period $x \sim \text{exponential}(\lambda = 3)$
 - Service time $y \sim \text{exponential}(\mu = 4)$



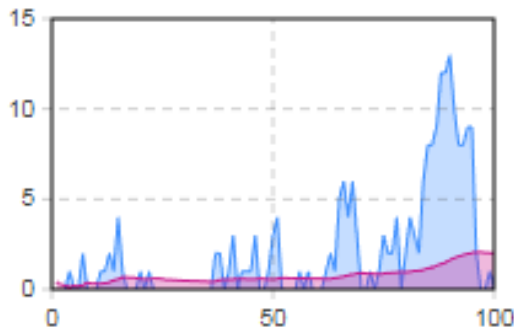


Queuing System Visualization

- Plots: `queue.size()` , `queue.statsSize.mean()`
- Histograms: `waitInQueueData` , `waitData`
 - Add agent parameter `enterSystem`
 - Log data using event actions:
 - Source on exit: `agent.enterSystem = time()` ;
 - Queue on exit:
`waitInQueueData.add(time() - agent.enterSystem)`
 - Sink on enter:
`waitData.add(time() - agent.enterSystem)`

Queuing System Outputs

Counts of enter/exit from each block

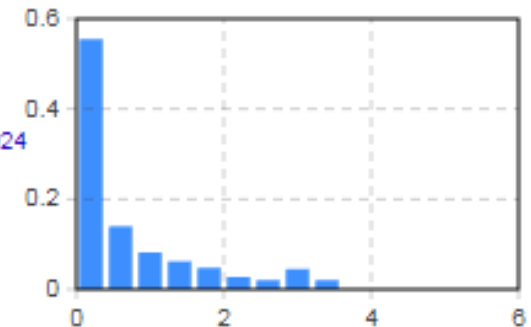


● Queue Length
● Average Queue Length

Time Series Plots

waitInQueueData
292 samples [0...3.58]. Mean=0.684

waitData
292 samples [3.31E-4...4.012]. Mean=0.924



● Wait Time in Queue 0.68
● Total Wait 0.92

Histograms



AnyLogic Example: Inventory Model





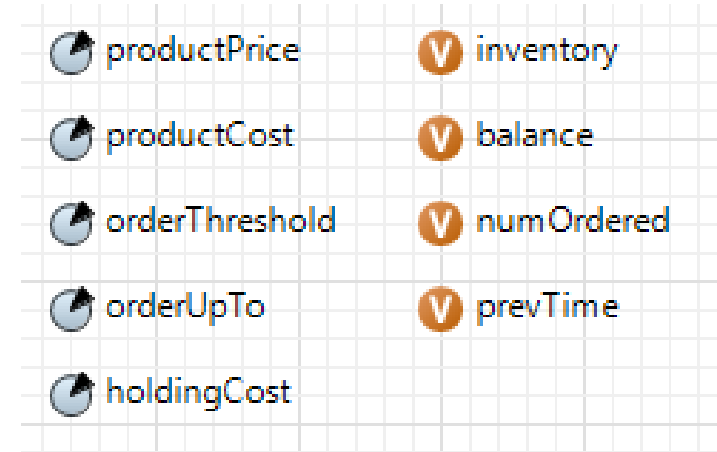
Inventory System Model

- Inventory system specification:
 - Sell products for $r = 100$ (can only sell those in stock)
 - Customer inter-arrival period $d \sim \text{exponential}(\lambda = 5)$
 - Each customer demands $D \sim \text{uniform}(1,4)$ products
 - Order policy (Q, S) : if inventory is $x < Q$, order $y = S - x$
 - Costs $c(y) = 50y$ to order y units
 - Delay of $L = 2$ days until delivery
 - Holding cost of $h = 2$ per item per day

Inventory System "Agent"

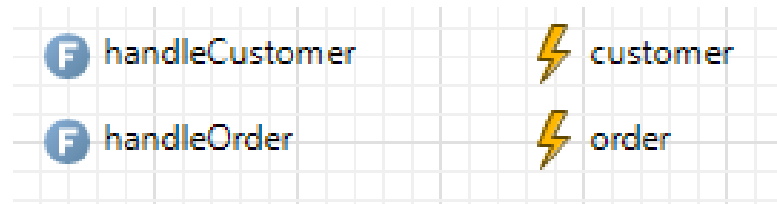
- Model state:

- Parameters:** product price, product cost, holding cost, order threshold (Q), order up to quantity (S)
- Variables:** Inventory, balance, number ordered, previous time



- Model behavior:

- Functions:** handle customer, handle order
- Events:** customer, order (both manually triggered)





Handle Customer Function

```
balance -= holdingCost*inventory*(time() - prevTime);
prevTime = time();
int demand = uniform_discr(1,4);
int numSold = 0;
if(inventory > demand) {
    numSold = demand;
} else {
    numSold = inventory;
}
balance += productPrice*numSold;
inventory -= numSold;
if(inventory < orderThreshold && numOrdered == 0) {
    numOrdered = orderUpTo - inventory;
    balance -= productCost*numOrdered;
    order.restart();
}
customer.restart(exponential(5));
```

Inventory System Outputs

productPrice
100

productCost
50

orderThreshold
10

orderUpTo
30

holdingCost
2

inventory
16

balance
34,829.512

numOrdered
0

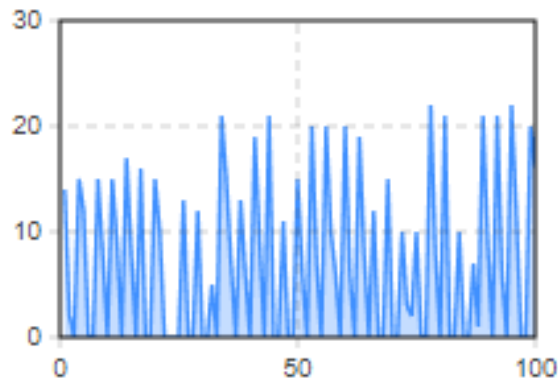
prevTime
99.684

handleCustomer

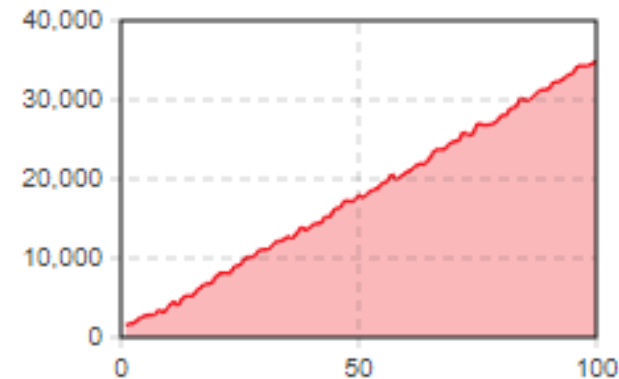
handleOrder

customer
0.223

order



● Inventory



● Balance



AnyLogic Example: Factory Model





Factory System Model



- Factory system specification:
 - $n = 50$ machines working 8 hours/day, 5 days/week
 - Each fails randomly after $d \sim \text{uniform}(132, 182)$ hours
 - Immediately replace with spare when available
 - Repairer fixes failed machine in $r \sim \text{uniform}(4, 10)$ hours, returns to spares
 - Hire R repairers at \$3.75/hour
 - Purchase S spares at \$30/day
 - Costs \$20/hour/machine if out-of-service (no spares)

Factory System "Agent"

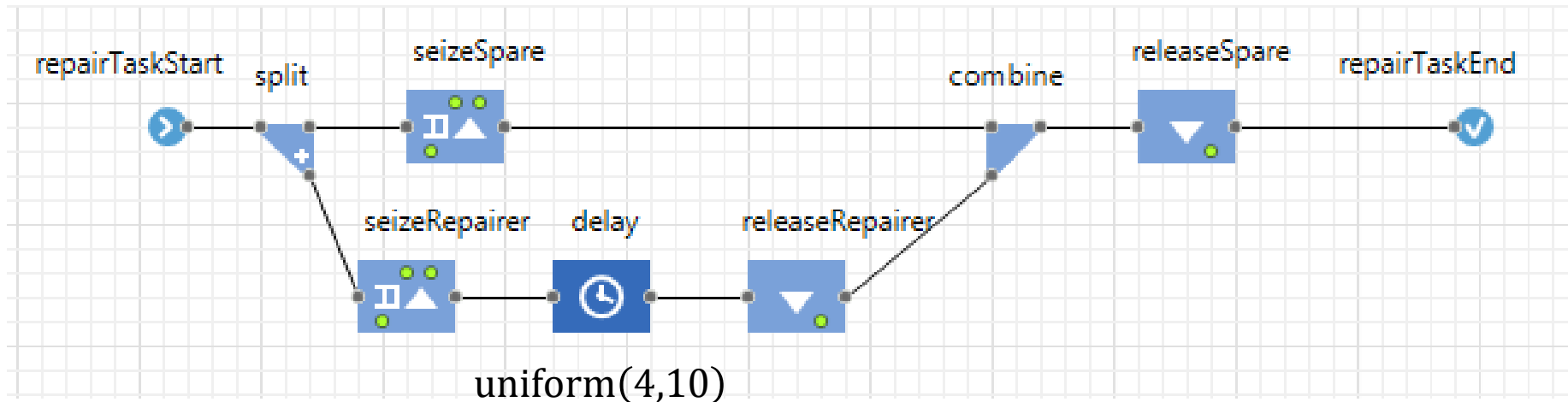
- Model state:
 - **Parameters:** number spares (S), number repairers (R), hourly wage, spares cost, out-of-service cost
 - **Variables:** total cost
- Model behavior:
 - **Function:** update daily cost
 - **Event:** end of day, triggers every 8 hours

 numSpares numRepairers repairerHourlyWage spareCostDay outOfServiceHourlyCost cost endOfDay updateDailyCost

Factory System "Process"

uniform(132,182)

- Resource pools:
 - Machines, spares, repairers
 - Machines have failure process with time between failures
- Machine pool failure flowchart
 - Tabulate cost of spare on "seizeSpare" process



Factory System Outputs



numSpares
20

numRepairers
5

repairerHourlyWage
3.75

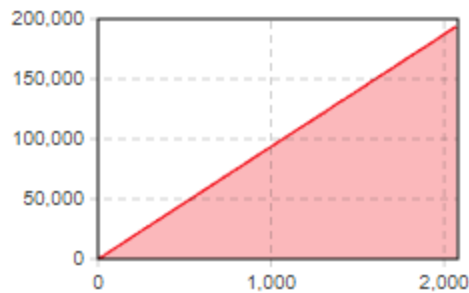
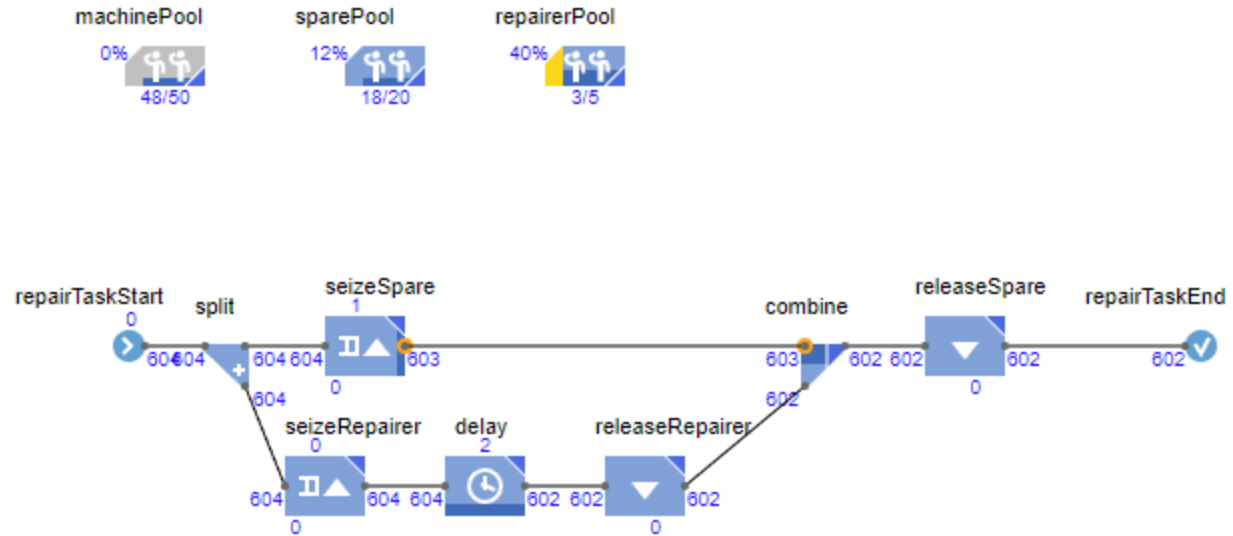
spareCostDay
30

outOfServiceHourlyCost
20

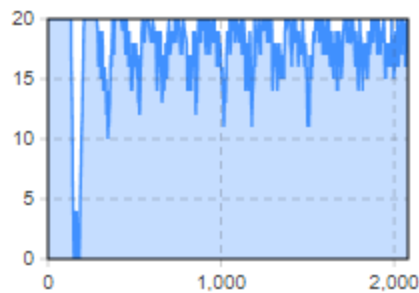
cost
195,093.74

endOfDay
8

updateDailyCost



● Total Cost



● Number Spares Available