

Lab 4: Simulation with Arena

Your task in this lab is to simulate and animate a model of a two-station queueing system in the simulation modeling environment Arena. Read the description below before starting the lab.

The Situation

Leviathan Limited, a very large enterprise indeed, maintains a repair facility that reconditions certain expensive products when they fail. The repair facility consists of a repair station, an inspection station and a combined repair-and-inspection station. The repair station makes the first attempt to repair a product. The product then moves to the inspection station to verify that the repairs have been successful. Products that pass inspection are shipped back to their owner. The small number of products that fail inspection move to the combined repair-and-inspection station where they are repeatedly repaired and inspected until they work properly. The original vision for this configuration was to exploit the efficiency of specialized repair and inspection stations for most products, and have a general-purpose station for the particularly difficult cases.

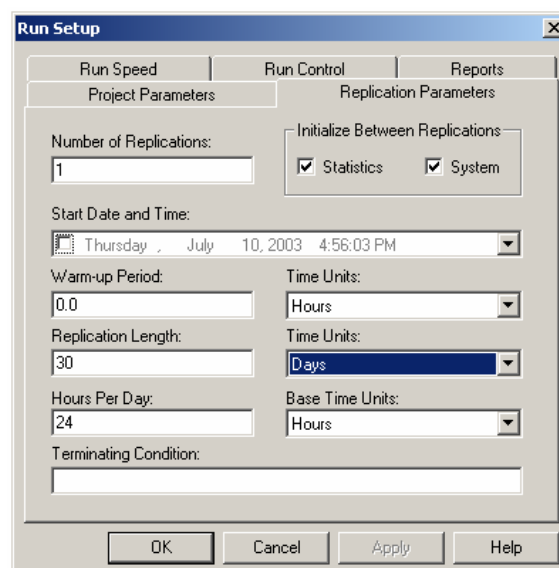
Regrettably, profits have not been good for Leviathan Ltd., so management is searching for ways to make more efficient use of resources. An industrial engineer (IE) has noticed that the combined repair-and-inspection station is not heavily utilized; she speculates that this station can be eliminated by sending products that fail inspection back to the primary repair station. For this to be a viable option it must not lead to excessive delays, so the IE must predict the impact of the change to support her proposal.

Some rough estimates are available. Products arrive at a rate of about 5 per day, can be repaired at a rate of about 6 per day, and are inspected at a rate of about 8 per day. Roughly 10 % of parts fail inspection, even if they have been repaired several time already. Simulate the proposed system for 30 days to estimate the average cycle time for repaired parts. Approximate all times as exponentially distributed.

Building the Model

The basic model is constructed as follows:

1. Using the Run:Setup... menu, set the Replication Length to 30 days, and set the Simulation Time Units to Days.



- Use a Create module to model a Poisson arrival process. Remember that in Arena Random(Expo) means exponential time between arrivals. Be sure to enter the *mean time between arrivals*, not the rate, and make sure the time unit is days.



Create

Name: Entity Type:

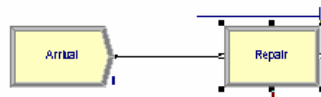
Time Between Arrivals

Type: Value: Units:

Entities per Arrival: Max Arrivals: First Creation:

OK Cancel Help

- Use a Process module to model the queue for repair. Make the name of this module Repair. Select Seize-Delay-Release as the action, and be sure to Add a resource to represent the repair person; give this resource the name Repair Person. For Delay Type choose Expression and use the function EXPO(Mean) to generate exponentially distributed service times. Be sure to specify the mean, not the rate, and make the time unit Days.



Resources

Type:

Resource Name: Quantity:

OK Cancel Help

Process

Name: Type:

Logic

Action: Priority:

Resources:

Add...
 Edit...
Delete

Delay Type: Units: Allocation:

Expression:

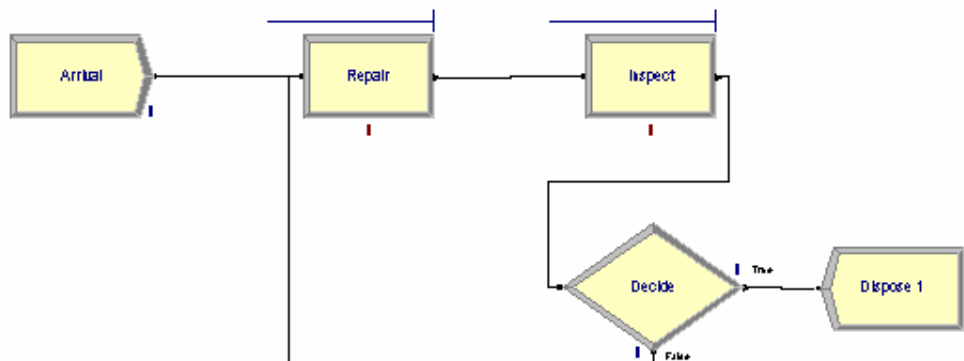
☒ Report Statistics

OK Cancel Help

- Use a Process module to model the queue for inspection. Make the name of this module Inspect. Select Seize-Delay-Release as the action, and be sure to Add a resource to represent the repair person; give this resource the name Inspector. For Delay Type choose Expression and use the function EXPO(Mean) to generate exponentially distributed service times. Be sure to specify the mean, not the rate, and make the time unit Days.

- Use a Decide module to model passing or failing inspection. Decide 2-way By Chance with a 90% chance of passing (being True). Connect the False outcome to the Repair module.

- Use a Dispose module to model repaired items leaving the system.




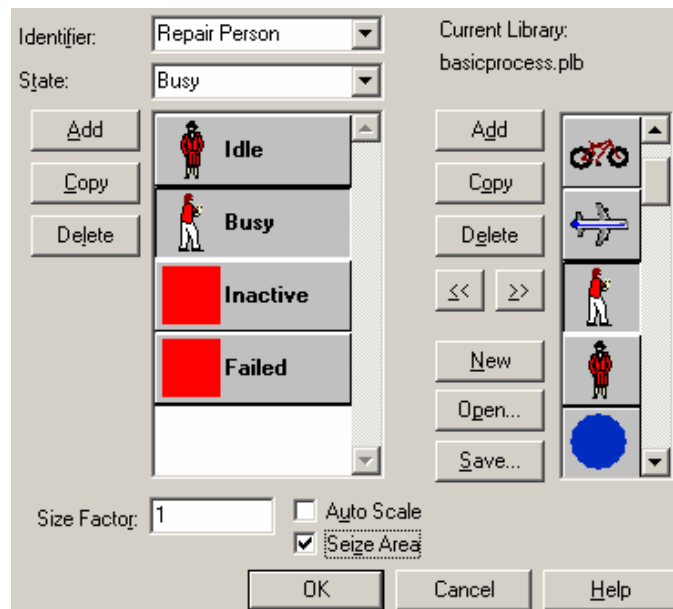
- Run the model and debug any errors that occur.



Once your model is running correctly, add additional animation as suggested below:

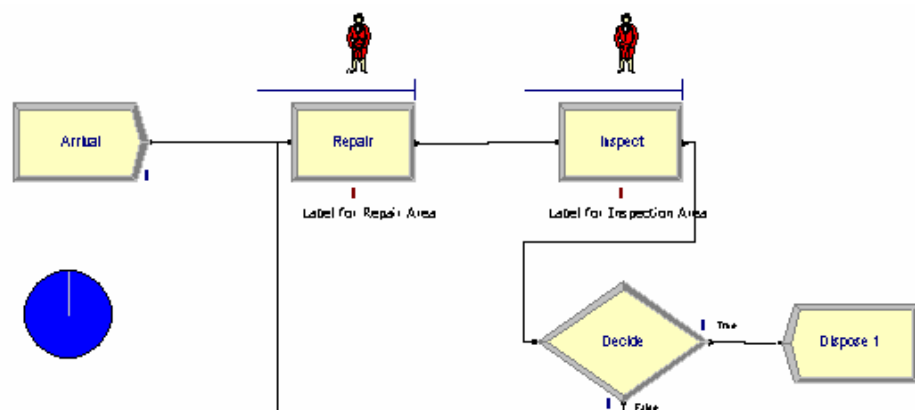
- Click on the Entity spreadsheet icon. Click on Initial Picture and use the drop-down menu to select a picture to represent your broken part entities.

Entity - Basic Process						
	Entity Type	Initial Picture	Holding Cost / Hour	Initial VA Cost	Initial NVA Cost	Initial Waiting Cost
1	Entity 1	Picture.Blue Ball	0.0	0.0	0.0	0.0
Double-click		Picture.Blue Ball				
		Picture.Blue Page				
		Picture.Boat				
		Picture.Box				
		Picture.Diskette				
		Picture.Email				
		Picture.Envelope				

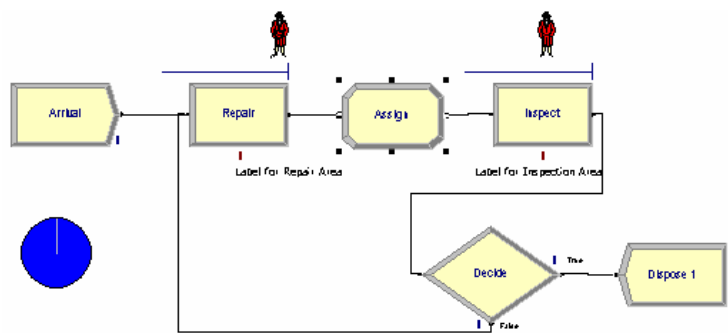
- From the Animate buttons, select Resource . This will open a resource dialog box. Do the following:
 - For the Identifier select Repair Person.
 - Select and copy pictures for the Idle and Busy states using the << key. You may have to open a picture library such as Workers.plb.
 - Be sure to select Seize Area; this causes the entity (broken item) to appear to be being worked on by the server.



- d. Click OK and place the resource picture near the Repair process. You can drag the corner of the picture to make it larger
3. Now create a similar resource picture for the Inspector.
4. From the Animate buttons, select Clock . Place it and drag to size.
5. Add labels to the repair and inspect areas using the text tool, . Add any other details or background colors you like by using the drawing tools.



6. Run your model and debug any errors.
7. Suppose we want to have the entity picture change after the item is repaired. To do this, place an Assign module between the two Process modules and connect them. Add the following assignment:
Type: Entity Picture
Entity Picture: pick something from the drop-down menu



Assign [?] [X]

Name: Assign

Assignments:

Entity Picture, Picture.Yellow Ball	Add...
<End of list>	Edit...
	Delete

OK Cancel Help

Assignments [?] [X]

Type: Entity Picture

Entity Picture: Picture.Yellow Ball

OK Cancel Help

Now when the entity passes through the Assign module its picture will change. Run your model to verify this.

What You Should Do

Now run the simulation for the full 30 days and observe the average cycle time statistic. Print out your model, write the average cycle time on it, and turn it in to Mr. Allen Lee by **12pm on March 14, 2005** at IS lab. **No late labs will be accepted!**

Appendix I: Using Category Overview Reports

Click Time

Average cycle time

Entity

Time

VA Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.01344246	(Insufficient)	0.00048295	0.04808769
NVA Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.00	(Insufficient)	0.00	0.00
Wait Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.00133767	(Insufficient)	0.00	0.03946526
Transfer Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.00	(Insufficient)	0.00	0.00
Other Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.00	(Insufficient)	0.00	0.00
Total Time	Average	Half Width	Minimum Value	Maximum Value
Entity 1	0.01478013	(Insufficient)	0.00048295	0.04915015

Other

Appendix II: Useful Arena Distribution Functions

Arena Function	Distribution
BETA(Alpha1, Alpha2)	Sample from beta distribution
ERLA(ExpMean, k)	Sample from Erlang distribution with k phases and mean k*ExpMean
EXPO(Mean)	Sample from exponential distribution
GAMM(a, b)	Sample from gamma distribution
LOGN(Mean, StdDev)	Sample from lognormal distribution
NORM(Mean, StdDev)	Sample from normal distribution
POIS(Mean)	Sample from Poisson distribution
TRIA(Min, Mode, Max)	Sample from triangular distribution
UNIF(Min, Max)	Sample from continuous uniform distribution
WEIB(a, b)	Sample from Weibull distribution