



Ahsanullah University of Science and Technology (AUST)
Department of Mechanical and Production Engineering

LABORATORY MANUAL

For the students of

Department of Mechanical and Production Engineering
3rd Year, 2nd Semester

Student Name :
Student ID :

**Department of Mechanical and Production Engineering
Ahsanullah University of Science and Technology (AUST)**

IPE 3216: Industrial Systems with Simulation Sessional

Credit Hour: 1.5

List of Experiments:

1. Simulation of Business Planning and Analysis Using Monte Carlo
2. Simulation of the newspaper seller problem.
3. Simulation of the repairing policy of bearings.
4. Simulation of an inventory management system with varying lead time and management policy.
5. Simulation of Car shop Counter Problem
6. Simulation of a Small Manufacturing System
7. Simulation and Visualization of a Restaurant Capacity Planning Problem
8. Simulation and visualization of bus scheduling and parking lot systems.
9. Simulation and visualization of a traffic system.
10. Simulation of work-force management by applying Agent-Based Modeling.
11. Project

Total Marks		
Assignment/Project	Attendance	Quiz
40	10	50

Experiment 1: Simulation of Business Planning and Analysis Using Monte Carlo

Objective

The objective of this experiment is to:

- Apply Monte Carlo simulation techniques to evaluate project feasibility by generating parallel outcomes based on probabilistic inputs.
- Simulate input variable uncertainty by modeling fluctuating Gross Revenue and Operating Costs simultaneously using distinct probability distributions.
- Quantify and visualize risk exposure by generating a final histogram to map the full distribution and standard deviation of potential net returns.

Software

- Matlab

Introduction

In traditional business planning, financial analysts often rely on single-point estimates to project future outcomes, such as assuming a fixed 8% revenue growth rate or static operating expenses. While these deterministic models are straightforward to build, they fail to account for real-world volatility—such as fluctuating market demand, supply chain disruptions, and unexpected competitor pricing shifts—leaving decision-makers blind to downside risks and potential worst-case scenarios. Monte Carlo simulation solves this fundamental vulnerability by replacing fixed, static numbers with mathematical probability distributions, such as Normal, Uniform, or Triangular curves. By executing thousands of random, independent computing trials using these distinct probability ranges, the simulation generates an entire statistical spectrum of parallel business outcomes rather than a single, unreliable "best guess" projection.

When integrated into comprehensive business analysis, this advanced technique transforms vague forecasting into a highly rigorous, data-driven science that quantifies uncertainty. Instead of guessing whether a new corporate venture or product launch will succeed, a Monte Carlo simulation provides executives with an exact statistical probability of breaking even or achieving a positive Net Present Value (NPV). It allows leadership teams to stress-test their operations under extreme simulated conditions, identify which volatile variables pose the greatest threat to profitability, and visually map out the full risk profile of a major investment before committing capital. Ultimately, this approach moves organizations away from "gut-feeling" decision-making, ensuring they are not just planning for a single ideal outcome, but are actively prepared for the unexpected.

Problem

A manufacturing company is considering purchasing a new piece of machinery for \$100,000 (Year 0 Investment). The machine has a useful life of 1 year. At the end of the year, the machine will be sold for a guaranteed scrap value of \$10,000.

The cash inflow generated by the machine during that 1 year is highly uncertain due to fluctuating market demand and variable operating costs.

You need to run a Monte Carlo simulation with 10,000 trials to determine the distribution of the project's Net Cash Flow at Year 1, and the probability that the project yields a total positive return (Total Inflows > Total Outflows).

Variables and Distributions

1. Initial Investment (Year 0 Outflow): \$100,000 (Fixed)
2. Scrap Value (Year 1 Inflow): \$10,000 (Fixed)
3. Gross Revenue (Year 1 Inflow): Modeled as a Normal Distribution with a mean (μ) of \$120,000 and a standard deviation (σ) of \$15,000.
4. Operating Costs (Year 1 Outflow): Modeled as a Uniform Distribution ranging between a minimum of \$20,000 and a maximum of \$40,000.

Code

```
% 1. Simulation Parameters
numTrials = 10000;
initialInvestment = 100000;
scrapValue = 10000;

% 2. Generate Random Variables
% Gross Revenue ~ N(120000, 15000^2)
meanRevenue = 120000;
stdDevRevenue = 15000;
grossRevenue = normrnd(meanRevenue, stdDevRevenue, [numTrials, 1]);

% Operating Costs ~ U(20000, 40000)
minCost = 20000;
```

```

maxCost = 40000;

operatingCosts = unifrnd(minCost, maxCost, [numTrials, 1]);

% 3. Calculate Deterministic Equation for Net Return
totalInflows = grossRevenue + scrapValue - operatingCosts;
netReturn = totalInflows - initialInvestment;

% 4. Statistical Analysis
meanNetReturn = mean(netReturn);
probabilityOfProfit = mean(netReturn > 0) * 100;
standardDeviationReturn = std(netReturn);

% 5. Output Display
fprintf('--- Simulation Summary Results ---\n');
fprintf('Expected Mean Net Return: $%.2f\n', meanNetReturn);
fprintf('Standard Deviation of Return: $%.2f\n', standardDeviationReturn);
fprintf('Probability of Positive Return: %.2f%%\n', probabilityOfProfit);

% 6. Plotting Results
figure;
histogram(netReturn, 50, 'EdgeColor', 'w');
grid on;
hold on;
xline(0, 'r-', 'LineWidth', 2, 'Label', 'Breakeven (0)');
title('Probability Distribution of Project Net Return');
xlabel('Net Return ($)');
ylabel('Frequency');

```

Experiment 2: Simulation of the newspaper seller problem.

Objectives

- To simulate the newspaper seller's inventory problem using random numbers and probability distributions under uncertain demand conditions.
- To evaluate and compare the profitability of different newspaper purchasing quantities in order to identify the optimal inventory decision.

Introduction

In many business and operational environments, decision-makers must determine an appropriate inventory level before the actual demand is known. Since demand can vary from one period to another, selecting an inventory quantity that maximizes profit while minimizing losses from excess stock or unmet demand becomes a challenging task. Traditional analytical methods may not adequately capture such uncertainty.

In this experiment, simulation is used to evaluate the profitability of different inventory levels under varying demand conditions. By comparing the average profits obtained from multiple simulated periods, the most suitable inventory quantity can be identified for decision-making purposes.

Problem: The Newspaper Seller's Problem

A paper seller buys the papers for 35 cents each and sells them for 50 cents each. Newspapers not sold at the end of the day are sold as scrap for 5 cents each. Newspapers can be purchased in the bundles of 10. Thus, the paper seller can buy 50, 60 and so on.

There are three types of newsdays "good," "fair" and "poor," with probabilities of 0.35, 0.45 and 0.20. The distribution of papers demanded on each of these days are given below:

Demand	Good	Fair	Poor
40	0.03	0.10	0.44
50	0.05	0.18	0.22
60	0.15	0.40	0.16
70	0.20	0.20	0.12
80	0.35	0.08	0.06
90	0.15	0.04	0.00
100	0.07	0.00	0.00

The profits are given by the following relationship:

profit = revenue from sales – cost of newspapers – lost profit from excess demand + salvage from sales of scrap papers.

The problem is to determine the optimum number of papers the seller should purchase. This will be accomplished by simulating demands for 20 day and recording profits from sales each day.

Random digit assignment for the type of newsday:

Type of Newsday	Probability	Cumulative probability	Random digit assignment
Good	0.35	0.35	01-35
Fair	0.45	0.80	36-80
Poor	0.20	1.00	81-100

Random digit assignment for newspapers demanded:

Demand	Cumulative Distribution			Random Digit Assignment		
	Good	Fair	Poor	Good	Fair	Poor
40	0.03	0.10	0.44	01–03	01–10	01–44
50	0.08	0.28	0.66	04–08	11–28	45–66
60	0.23	0.68	0.82	09–23	29–68	67–82
70	0.43	0.88	0.94	24–43	68–88	83–94
80	0.78	0.96	1.00	44–78	89–96	95–00
90	0.93	1.00	1.00	79–93	97–00	
100	1.00	1.00	1.00	94–00		

(The ranges ending in 00 represent the random numbers up to 100, e.g., 94-100 or 97-100.)

MATLAB Code

```
total_days = 20;
days = (1:total_days)';
buying_price = 0.33;
selling_price = 0.50;
scrap_price = 0.05;

newspaper_type_probability = [0.35; 0.45; 0.20];
demands_available = (40:10:100)';

rand_digits_good = [3; 8; 23; 43; 78; 93; 100];
rand_digits_fair = [10; 28; 68; 88; 96; 100];
rand_digits_poor = [44; 66; 82; 94; 100];

per_day_demands = zeros(total_days, 1);
newspaper_types = days(discretetime(newspaper_type_probability, total_days));

per_sim_profits = zeros(7, 1);
simulation = 1;

for i = 1:total_days
    rand_digit = unidrnd(100);

    if newspaper_types(i) == 1
        for j = 1:7
            if rand_digit <= rand_digits_good(j)
                per_day_demands(i) = demands_available(j);
                break;
            end
        end
    end
end

end

end

end

for bought = (40:10:100)
    per_day_revenue = zeros(total_days, 1);
    cost_of_newspaper = zeros(total_days, 1);
    per_day_lost_profit = zeros(total_days, 1);
    per_day_salvage = zeros(total_days, 1);
    per_day_profit = zeros(total_days, 1);

    for i = 1:total_days
        if bought < per_day_demands(i)
            per_day_revenue(i) = bought * selling_price;
            per_day_lost_profit(i) = (per_day_demands(i) - bought) * (selling_price - buying_price);
            per_day_salvage(i) = 0;
        else
            per_day_revenue(i) = per_day_demands(i) * selling_price;
            per_day_lost_profit(i) = 0;
            per_day_salvage(i) = (bought - per_day_demands(i)) * scrap_price;
        end
    end
end
```

```

cost_of_newspaper(i) = buying_price * bought;
    per_day_profit(i) = per_day_revenue(i) - cost_of_newspaper(i) -
per_day_lost_profit(i) - per_day_salvage(i);
    end

    for i = 1:total_days
        if bought < per_day_demands(i)
            per_day_revenue(i) = bought * selling_price;
            per_day_lost_profit(i) = (per_day_demands(i) - bought) * (selling_price
- buying_price);
            per_day_salvage(i) = 0;
        else
            per_day_revenue(i) = per_day_demands(i) * selling_price;
            per_day_lost_profit(i) = 0;
            per_day_salvage(i) = (bought - per_day_demands(i)) * scrap_price;
        end

        cost_of_newspaper(i) = buying_price * bought;
        per_day_profit(i) = per_day_revenue(i) - cost_of_newspaper(i) -
per_day_lost_profit(i) - per_day_salvage(i);
        end

        per_sim_profits(simulation) = sum(per_day_profit);
        simulation = simulation + 1;
    end

    [max_profit, max_profit_sim] = max(per_sim_profits);

    final_result = [max_profit, max_profit_sim];
    final_names = {'max_profit', 'max_profit_sim'};
    final_result_table = array2table(final_result, 'variablenames', final_names);

```

Experiment 3: Simulation of the repairing policy of bearings.

Objective

- To simulate the failure and replacement process of bearings in a milling machine using MATLAB and evaluate system reliability over a specified operating period.
- To compare the total maintenance cost, downtime cost, and replacement cost under reactive and preventive maintenance policies to determine an economical strategy.

Introduction

Reliability engineering focuses on analyzing the performance and failure behavior of machine components over time. Unexpected equipment failures can cause production interruptions, increase maintenance expenses, and reduce overall operational efficiency. Since such failures occur randomly, simulation techniques are widely used to model system behavior and evaluate different maintenance strategies before implementing them in real-world operations.

In this experiment, MATLAB is used to simulate the operation of a milling machine containing three bearings. The simulation incorporates bearing lifetimes, repairperson delays, replacement costs, labor costs, and machine downtime costs. The results are then used to compare reactive and preventive maintenance policies and determine the most cost-effective maintenance approach.

Problem:

A large milling machine has three different bearings that fail in service. The cumulative distribution function of the life of each bearing is identical as shown in Table 1. When a bearing fails, the mill stops, a repairperson is called, and a new bearing is installed. The delay time of the repairperson's arriving at the milling machine is also a random variable, with the distribution given in Table 2. Downtime for the mill is estimated at \$5 per minute. The direct on-site cost of the repairperson is \$15 per hour. It takes 20 minutes to change one bearing, 30 minutes to change two bearings, and 40 minutes to change three bearings. The bearings cost \$16 each. A proposal has been made to replace all three bearings whenever a bearing fails. Management need an evaluation of this proposal.

Table 1: Bearing-Life Distribution

Bearing Life (Hours)	Probability	Cumulative Probability	Random Digit Assignment
1000	0.10	0.10	01-10
1100	0.13	0.23	11-23
1200	0.25	0.48	24-48
1300	0.13	0.61	49-61
1400	0.09	0.70	62-70
1500	0.12	0.82	71-82
1600	0.02	0.84	83-84
1700	0.06	0.90	85-90
1800	0.05	0.95	91-95
1900	0.05	1.00	96-00

Table 2: Delay-Time Distribution

Delay Time (Minutes)	Probability	Cumulative Probability	Random Digit Assignment
5	0.6	0.6	1-6
10	0.3	0.9	7-9
15	0.1	1.0	0

Simulate the problem for 20000 hours of operation. It will be assumed in this problem that the times are never exactly the same and thus no more than 1 bearing is changed at any breakdown. 16 bearing changes were made for bearings 1 and 2 but only 14 bearing changes were required for bearing 3.

MATLAB Code:

```
% Parameters
repairperson_cost = 15 / 60;
bearing_cost = 16;
max_operation_time = 20000;

bearing_life_hours = (1000:100:1900)';
bearing_life_probability = [0.10; 0.13; 0.25; 0.13; 0.09; 0.12; 0.02; 0.06; 0.05;
0.05];

delay_time_minutes = (5:5:15)';
delay_time_probability = [0.6; 0.3; 0.1];

% First case
per_bearing_life_sum = zeros(3, 1);
per_delay_time_sum = zeros(3, 1);
per_replacement_cost = zeros(3, 1);
per_repairperson_cost = zeros(3, 1);
per_downtime_cost = zeros(3, 1);
per_bearing_total_cost = zeros(3, 1);
```

```

for i = 1:3
    no_of_bearing = 0;

    for j = 1:20
        bearing_life = bearing_life_hours(discretesample(bearing_life_probability,
1));
        per_bearing_life_sum(i) = per_bearing_life_sum(i) + bearing_life;

        if per_bearing_life_sum(i) >= max_operation_time
            no_of_bearing = j;
            break;
        end

        delay_time = delay_time_minutes(discretesample(delay_time_probability, 1));
        per_delay_time_sum(i) = per_delay_time_sum(i) + delay_time;
    end

    per_replacement_cost(i) = bearing_cost * (no_of_bearing - 1);
    per_repairperson_cost(i) = repairperson_cost * 20 * (no_of_bearing - 1);
    per_downtime_cost(i) = (delay_time_sum(i) + (no_of_bearing - 1) * 20) *
downtime_cost;
    per_bearing_total_cost(i) = per_replacement_cost(i) + per_repairperson_cost(i)
+ per_downtime_cost(i);
end

total_cost = sum(per_bearing_total_cost);

% Second case
bearing_life_sum_case2 = 0;
delay_time_sum_case2 = 0;
replacement_cost_case2 = 0;
repairperson_cost_case2 = 0;
downtime_cost_case2 = 0;
total_cost_case2 = 0;

for i = 1:20
    no_of_bearing = 0;

    per_bearing_life = bearing_life_hours(discretesample(bearing_life_probability,
3));
    min_bearing_life = min(per_bearing_life);
    bearing_life_sum_case2 = bearing_life_sum_case2 + min_bearing_life;

    if bearing_life_sum_case2 >= max_operation_time
        no_of_bearing = i;
        break;
    end

    delay_time = delay_time_minutes(discretesample(delay_time_probability, 1));
    delay_time_sum_case2 = delay_time_sum_case2 + delay_time;
end

replacement_cost_case2 = 3 * bearing_cost * (no_of_bearing - 1);
repairperson_cost_case2 = repairperson_cost * 40 * (no_of_bearing - 1);

```

```
downtime_cost_case2 = (delay_time_sum_case2(i) + (no_of_bearing - 1) * 40) *  
downtime_cost;  
total_cost_case2 = replacement_cost_case2 + repairperson_cost_case2 +  
downtime_cost_case2;  
  
% Compare and choose policy  
if total_cost > total_cost_case2  
    fprintf('Choose second replacement policy\n');  
else  
    fprintf('Choose first replacement policy\n');  
end
```

Experiment 4: Simulation of an inventory management system with varying lead time and management policy.

Objectives

- **Simulate System Dynamics:** To model a periodic review inventory management system over five cycles utilizing random digit assignments for variable daily demands and order lead times.
- **Evaluate Performance Metrics:** To estimate the average ending units in inventory and determine the exact number of days a shortage condition occurs under the specified management policy.

Software

- Matlab

Introduction

Effective inventory management is a critical component of supply chain operations, requiring a delicate balance between minimizing holding costs and preventing customer stockouts. This experiment addresses this operational challenge by simulating a periodic review inventory system governed by an (M, N) policy, where stock levels are reviewed every 5 days and replenished up to a maximum capacity of 11 units. Because real-world conditions are rarely predictable, the simulation incorporates random digit assignments to model the inherent uncertainty in both daily customer demand and supplier lead times based on provided probability distributions. By tracking the system's daily fluctuations over five distinct cycles, this computational approach provides a reliable method to estimate average ending inventory and evaluate the frequency of shortages, ultimately offering actionable insights into the robustness of the current management strategy.

Problem

Suppose that the maximum inventory level M is 11 units and the review period N is 5 days. The problem is to estimate by simulation (for 5 cycles) the average ending units in inventory and the number of days when a shortage condition existed. The distribution of the number of units demanded per day is shown in table. Lead time is random variable also shown in table. Assume that the orders are placed at the close of business.

Table 1: Random Digit Assignments for Daily Demand

Demand	Probability	Cumulative Probability	Random Digit Assignment
0	0.10	0.10	01-10
1	0.25	0.35	11-35
2	0.35	0.70	36-70
3	0.21	0.91	71-91
4	0.09	1.00	92-100

Table 2: Random Digit Assignments for Lead Time

Lead Time (Days)	Probability	Cumulative Probability	Random Digit Assignment
1	0.6	0.6	1-6
2	0.3	0.9	7-9
3	0.1	1.0	0

Solution

```
inventory.m* x discretetample.m x +
1 - M = 11;
2 - N = 5;
3 - total_cycle = 5;
4 - total_days = N * total_cycle;
5
6 - days = (1:total_days)';
7 - begin_inventory = [3; zeros(total_days - 1, 1)];
8 - end_inventory = zeros(total_days, 1);
9
10 - demands_list = (0:4)';
11 - demands_probability = [0.10, 0.25, 0.35, 0.21, 0.09]';
12 - per_day_demands = demands_list(discretetample(demands_probability, total_days));
13
14 - lead_time_list = (1:3)';
15 - lead_time_probability = [0.6, 0.3, 0.1]';
16 - per_cycle_lead_time = lead_time_list(discretetample(lead_time_probability, total_cycle));
17 - lead_time = zeros(total_days, 1);
18
19 - order = zeros(total_days, 1);
20 - shortage = zeros(total_days, 1);
21
22 - j = 1;
23 - for i = N:N:total_days
24 -     lead_time(i) = per_cycle_lead_time(j);
25 -     j = j + 1;
26 - end
27
28 - for i = 1:total_days
29 -     if i == 2
30 -         begin_inventory(i) = begin_inventory(i) + 8;
31 -     end
32
33 -     if begin_inventory(i) >= per_day_demands(i)
34 -         end_inventory(i) = begin_inventory(i) - per_day_demands(i);
35 -         shortage(i) = 0;
36
37 -         if i + 1 <= total_days
38 -             begin_inventory(i + 1) = begin_inventory(i + 1) + end_inventory(i);
39 -         end
40 -     else
41 -         end_inventory(i) = 0;
42 -         shortage(i) = per_day_demands(i) - begin_inventory(i);
43
44 -         if i + 1 <= total_days
45 -             per_day_demands(i + 1) = per_day_demands(i + 1) + shortage(i);
46 -         end
47 -     end
48
49 -     if rem(i, N) == 0
50 -         order(i) = M - end_inventory(i);
51 -     end
52
53 -     if (order(i) > 0) && (i + lead_time(i) <= total_days)
54 -         begin_inventory(i + lead_time(i)) = begin_inventory(i + lead_time(i)) + order(i);
55 -     end
56 - end
57
58 - average_ending_units = mean(end_inventory);
59 - num_of_shortage_days = nnz(shortage);
60
61 - result = [days, begin_inventory, per_day_demands, end_inventory, shortage, order, lead_time];
62 - names = {'days', 'begin_inventory', 'demands', 'end_inventory', 'shortage', 'order', 'lead_time'};
```

Assignment

Rework the above example for 10 cycles with $M = 10$

Experiment 5: Simulation of Car shop Counter Problem

Objective

The objective of this experiment is to:

- Construct and simulate a car shop counter system to accurately track customer flow
- Evaluate how a single processing resource behaves when subjected to disparate, non-exponential service time distributions (such as Uniform and Triangular distributions) running in parallel.
- Measure key performance indicators (KPIs)—including instantaneous resource utilization, average value-added (VA) processing times, and average entity waiting times—to pinpoint processing bottlenecks over a fixed temporal window.

Software

- Rockwell Arena Simulation/AnyLogic

Introduction

In industrial engineering and operations research, queuing systems represent a foundational framework for analyzing the flow of entities through constrained processes. Real-world service systems are rarely uniform; they are characterized by stochastic (random) arrivals and variable processing times. When a single resource must handle multiple classes of arrivals, managing congestion becomes a complex balancing act between resource efficiency and customer satisfaction. Discrete-event simulation allows engineers to model these unpredictable environments to analyze system capacity and performance dynamically over time.

A critical challenge in managing queuing networks is handling system constraints, specifically finite capacity and human behavior. When physical space or safety limits restrict a queue's length, it introduces the concept of balking—a phenomenon where an arriving entity refuses to join the line because the wait or congestion is visibly too high. Modeling these operational "gates" is vital because capacity limits don't just create longer lines; they directly cause system rejection and lost throughput.

Furthermore, when a single server is cross-trained to handle multiple distinct tasks, the system achieves operational flexibility but introduces high variance in processing times. By utilizing software packages like Rockwell Arena, complex operational environments can be modeled as high-fidelity digital twins. This computational approach allows for the stress-testing of strict boundary conditions, enabling engineers to quantitatively evaluate the trade-offs between resource utilization and entity loss. Ultimately, analyzing a system through this simulated lens provides critical data-driven insights necessary to optimize infrastructure constraints, stabilize service delivery speeds, and make informed decisions on capacity expansion without incurring real-world operational costs.

Problem

A local car repair and parts shop wants to analyze its front counter operations during an 8-hour workday. Customers arrive at the shop following an exponential distribution with a mean inter-arrival time of 1.5 minutes.

Due to limited physical space, the shop has a strict capacity rule: if there are already 6 customers waiting in the counter line, any newly arriving customer will immediately leave (balk) and go to a competitor.

Customers who successfully enter the shop are split into two categories:

- 70% are here for a Service Drop-off, which takes a triangular processing time of $\text{TRIA}(3, 5, 8)$ minutes.
- 30% are here as Parts Walk-ins, which takes a uniform processing time of $\text{UNIF}(1, 4)$ minutes.

Both types of customers are served sequentially by a single Counter Attendant resource on a first-come, first-served basis.

Build a discrete-event simulation model in Rockwell Arena to track system performance over 480 minutes. Run the simulation for 1 replication and report the following performance metrics:

1. Total number of customers who successfully entered the system.
2. Total number of customers who balked due to capacity.
3. The average value-added time, waiting time, and total time spent in the system per customer.
4. The utilization rate of the counter attendant.

Model

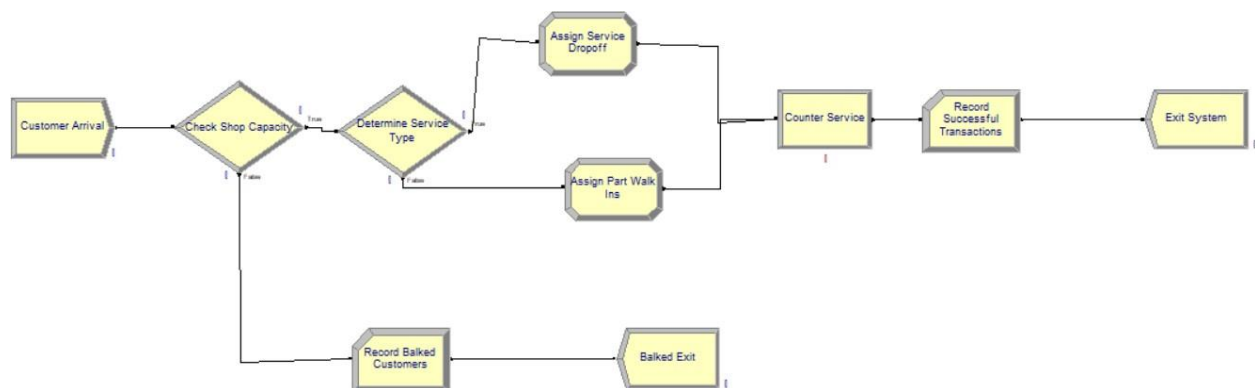
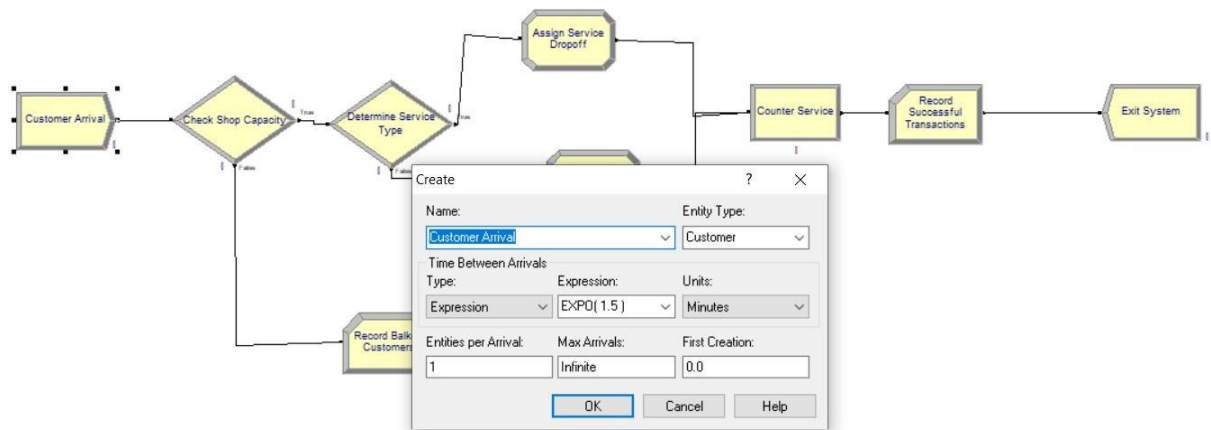


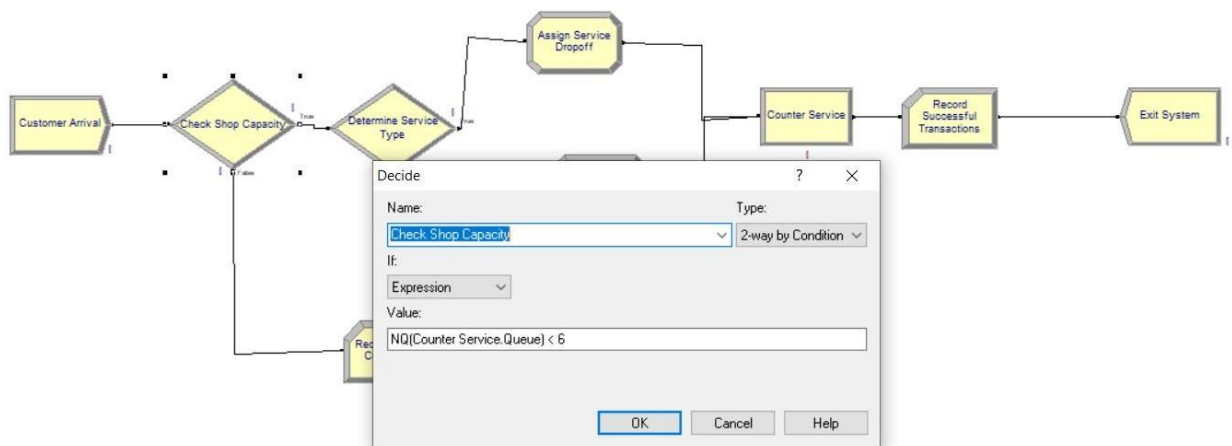
Figure 5.1: Simulation Model of Car Shop Counter

Configurations

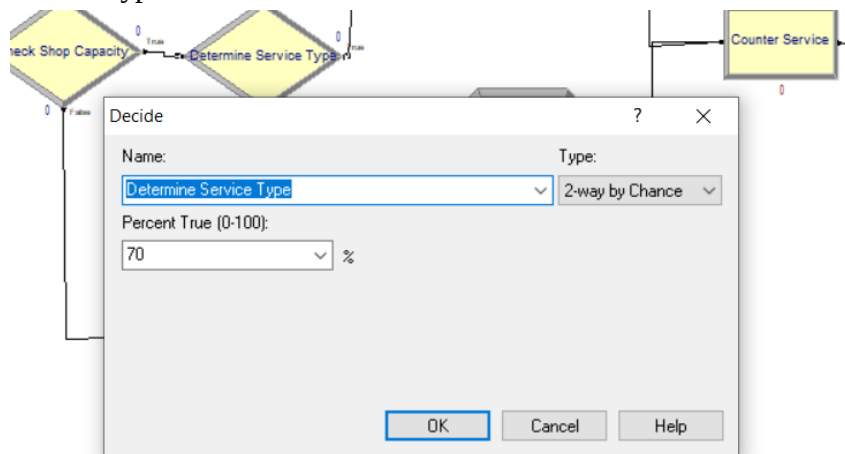
1. Customer Arrival



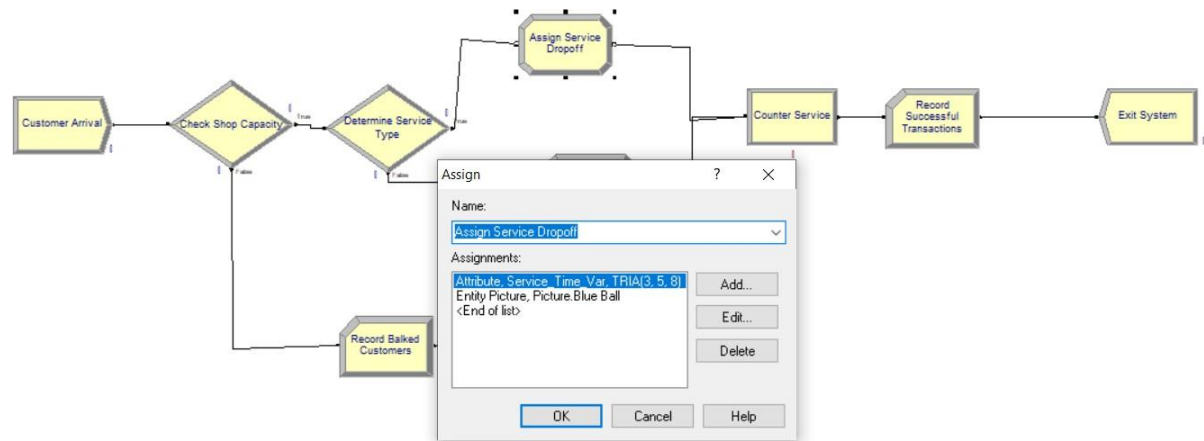
2. Check Shop Capacity



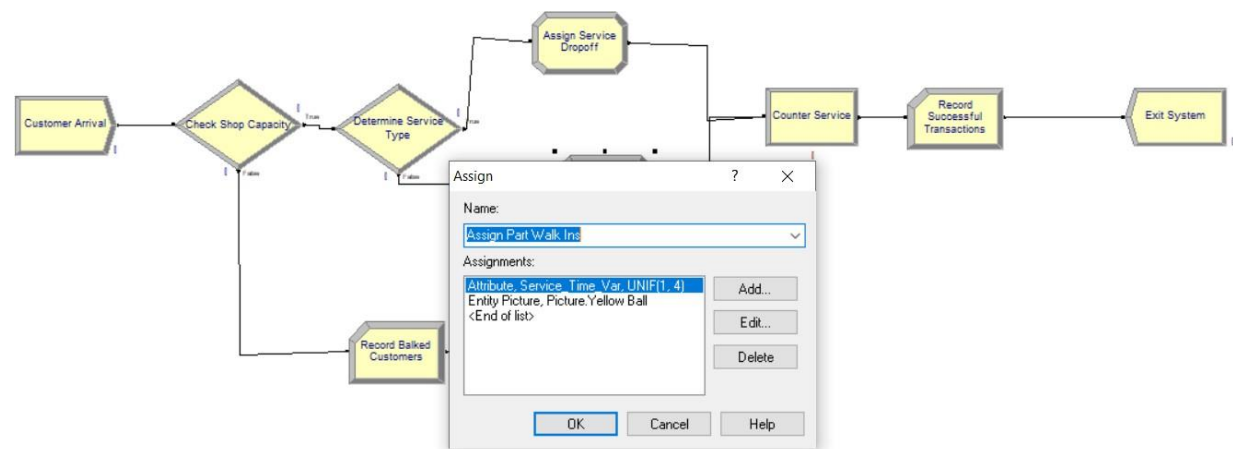
3. Determine Service Type



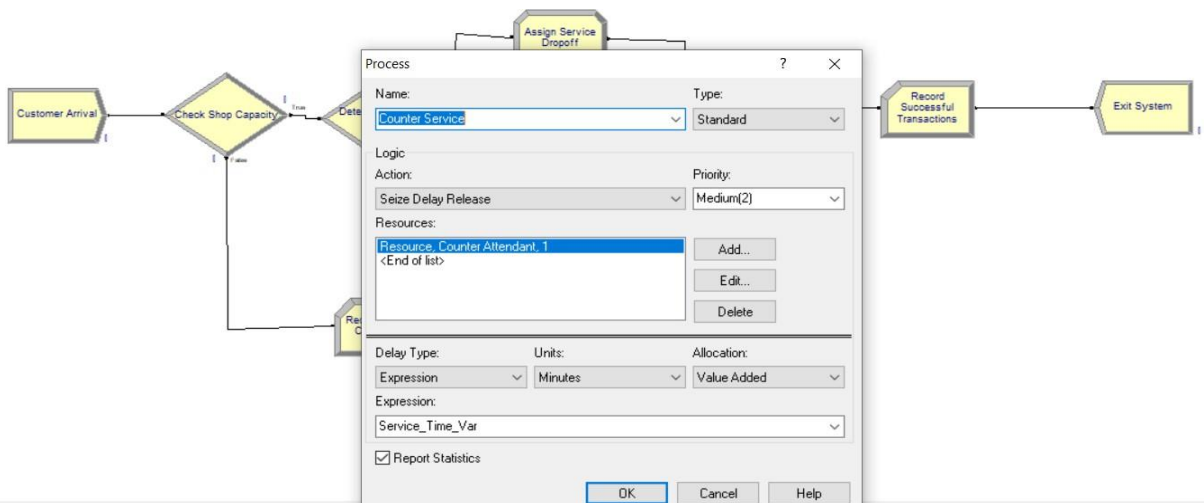
4. Assign Service Drop Off



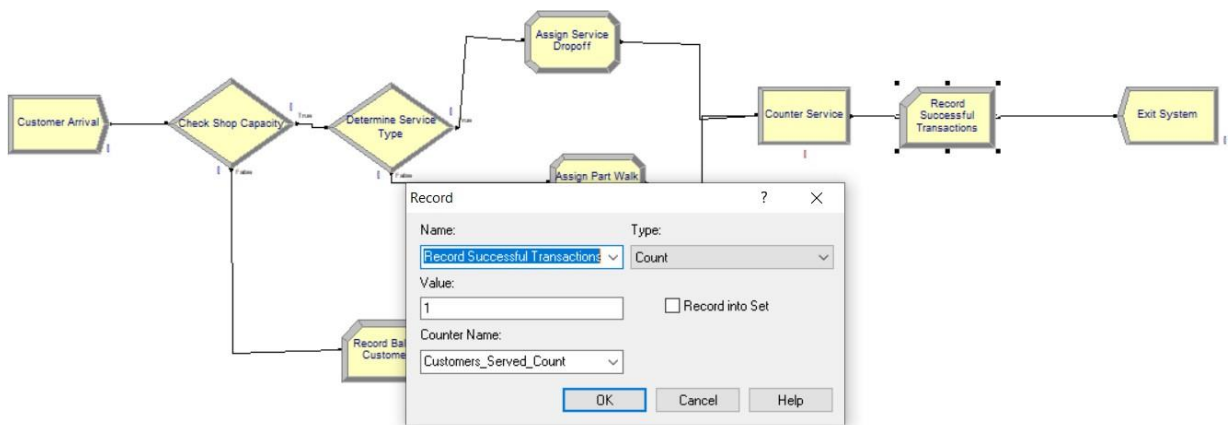
5. Assign Part Walk In



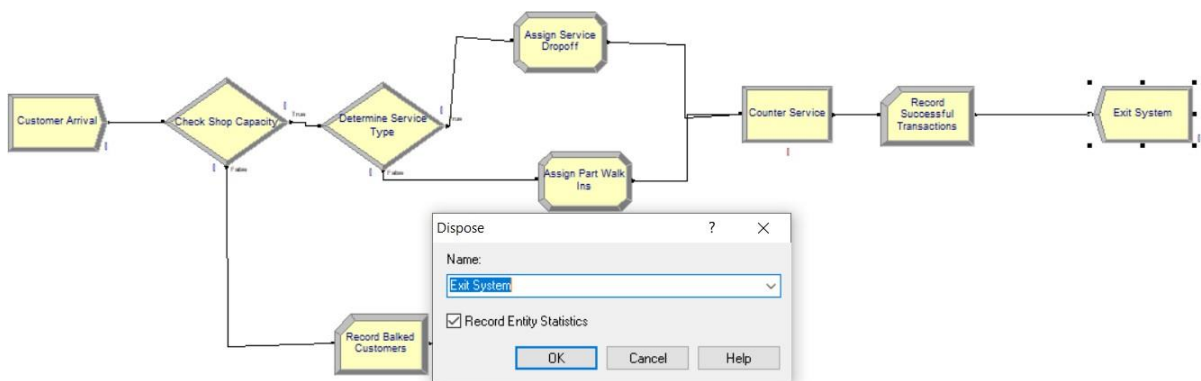
6. Counter Service



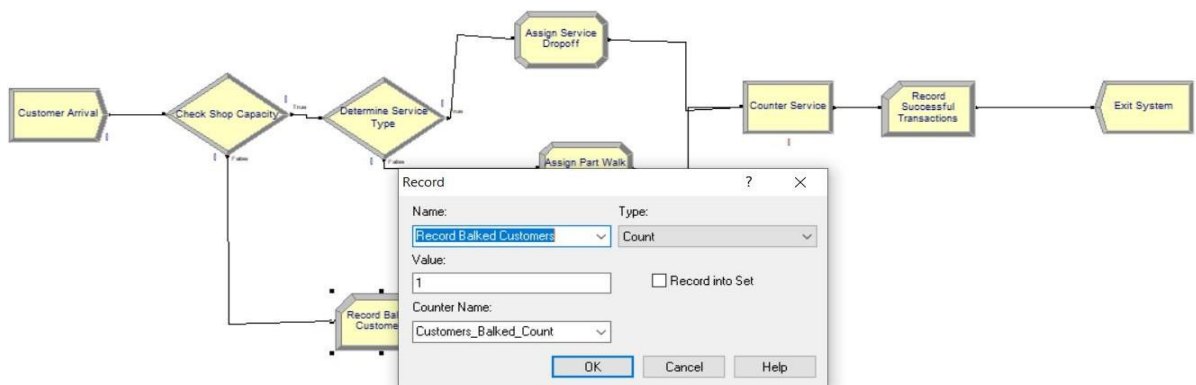
7. Record Successful Transactions



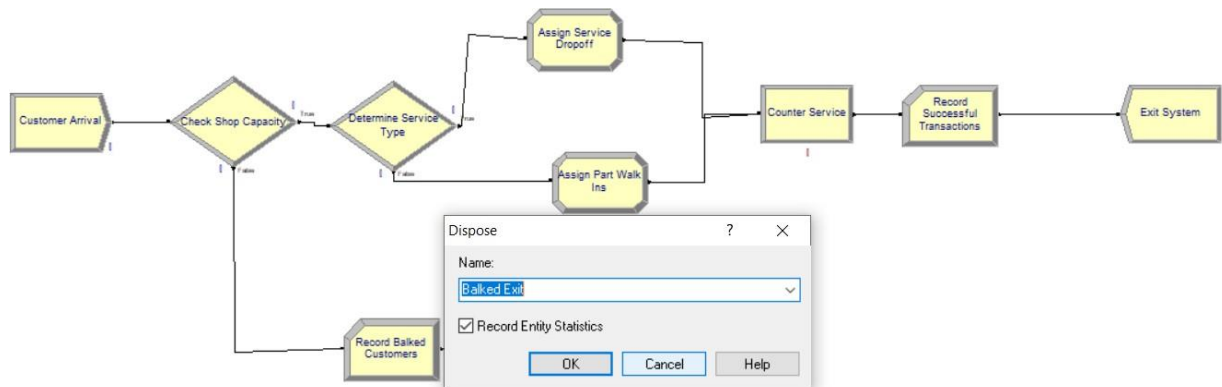
8. Exit System



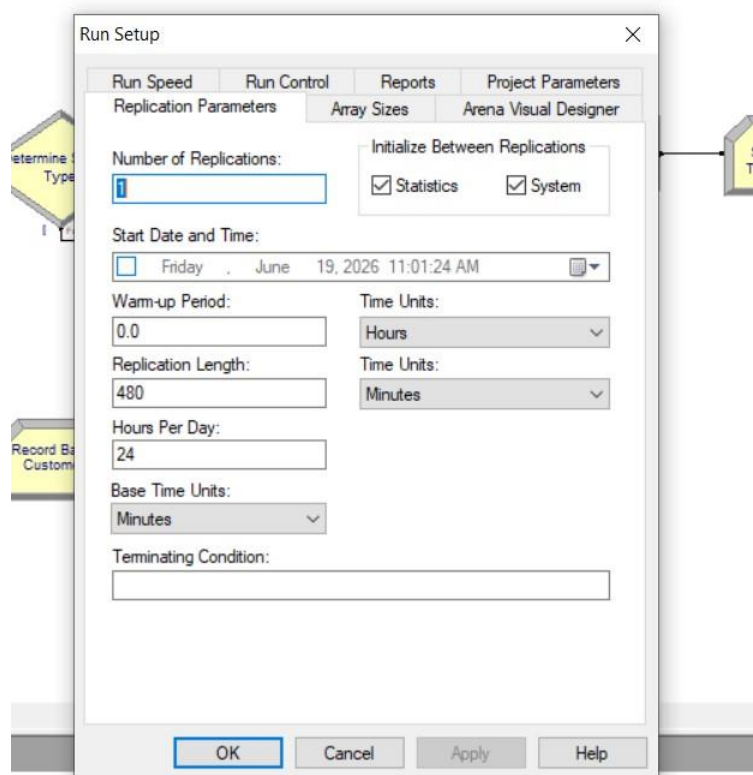
9. Record Bulk Customers



10. Balked Exit



Run Set Up



Experiment 6: Simulation of a Small Manufacturing System

Objectives

1. To develop a simulation model of a multi-product manufacturing system using Arena's advanced transfer modules.
2. To understand the application of Sequences, Stations, Routes, Resource Sets, Variables, Expressions, and Attributes in manufacturing simulation.
3. To analyze the performance of a manufacturing system producing multiple part types with different routing plans and processing requirements.
4. To evaluate machine utilization, queue statistics, throughput, and cycle time under realistic production conditions.

Theory

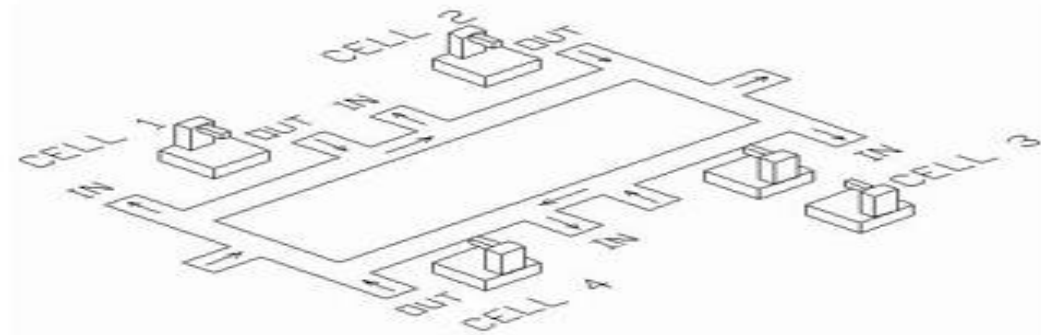
Manufacturing systems frequently produce multiple product types that follow different processing routes through a collection of machines and workstations. Modeling such systems requires mechanisms that can automatically route parts based on their product type while accounting for resource constraints, transfer delays, and varying processing times. Arena provides advanced modeling capabilities through Sequences, Stations, Routes, Resource Sets, Variables, Expressions, and Attributes that enable the construction of detailed manufacturing models. In this experiment, three-part types travel through four manufacturing cells according to predefined process plans. Cell 3 contains two non-identical machines, one of which operates 20% faster than the other. The simulation model captures the interaction between part arrivals, machine processing, routing decisions, and material movement throughout the facility. Performance measures such as machine utilization, average queue length, waiting time, throughput, and cycle time are collected to assess system performance and identify operational bottlenecks.

Problem Statement

A manufacturing facility consists of four production cells. Cells 1, 2, and 4 each contain one machine, while Cell 3 contains two machines. One machine in Cell 3 is a newer model that can process parts in only 80% of the time required by the older machine.

Three different part types are produced in the system. Each part follows a unique routing sequence through the manufacturing cells. Parts arrive according to an exponential distribution with a mean interarrival time of 13 minutes, with the first arrival occurring at time zero. The distribution by type is 26%, Part 1; 48%, Part 2; and 26%, Part 3. Parts enter from the left, exit at the right, and move only in a clockwise direction through the system. For now, we'll also assume that the time to move between any pair of cells is 2 minutes, regardless of the distance (we'll fix this up later). We want to collect statistics on resource utilization, time and number in queue, as well as cycle

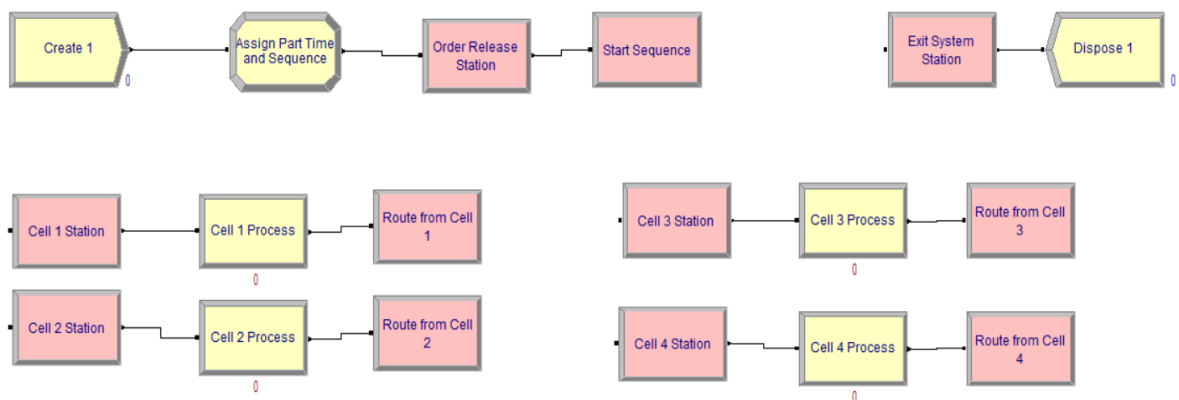
time (time in system, from entry to exit) by part type. Initially, we'll run our simulation for 32 hours.



Part Routing and Process Times

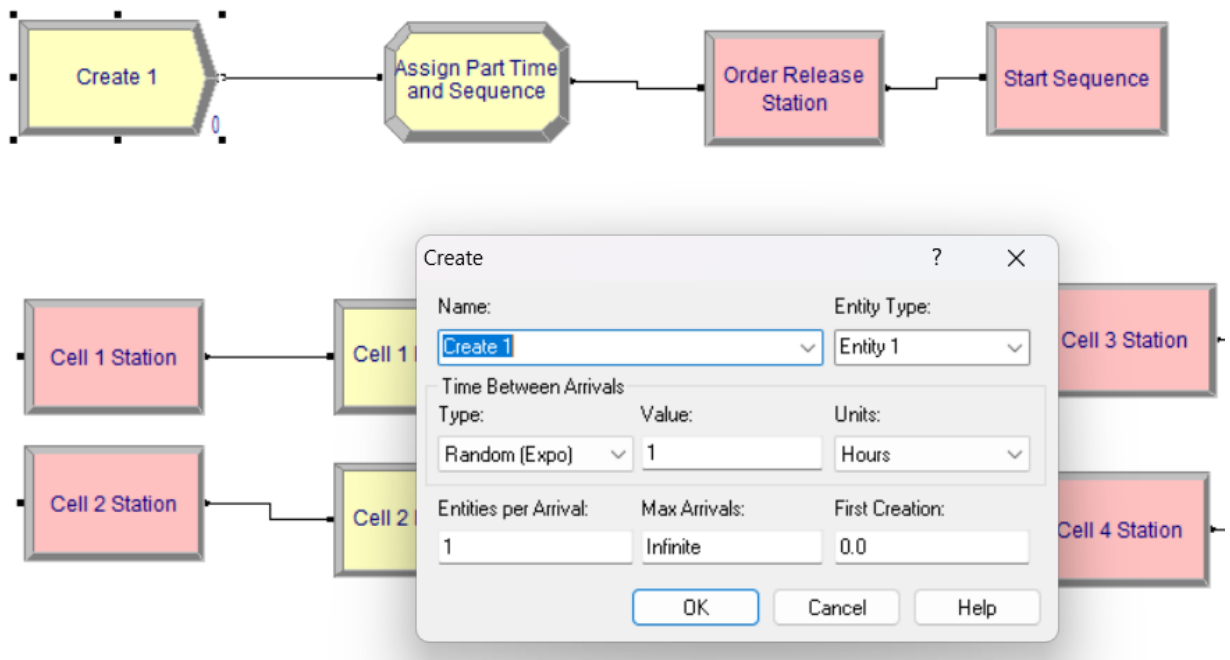
Part Type	Cell/Time	Cell/Time	Cell/Time	Cell/Time	Cell/Time
1	1	2	3	4	
	6, 8, 10	5, 8, 10	15, 20, 25	8, 12, 16	
2	1	2	4	2	3
	11, 13, 15	4, 6, 8	15, 18, 21	6, 9, 12	27, 33, 39
3	2	1	3		
	7, 9, 11	7, 10, 13	18, 23, 28		

Model

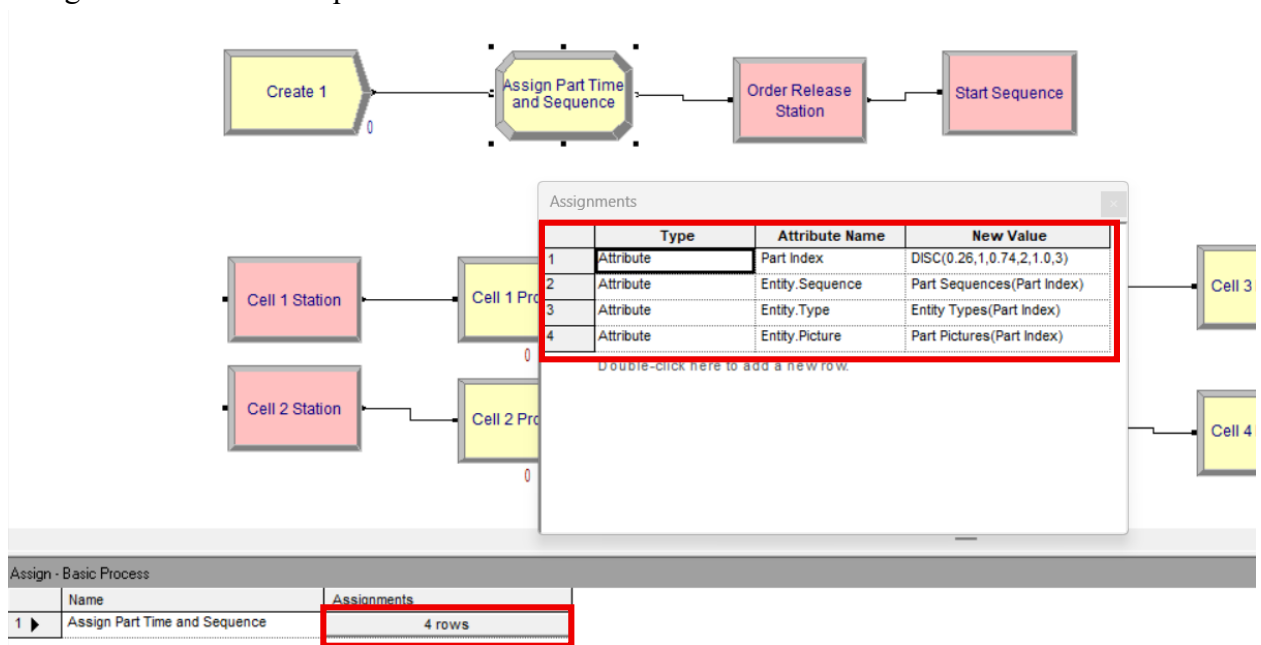


Configurations

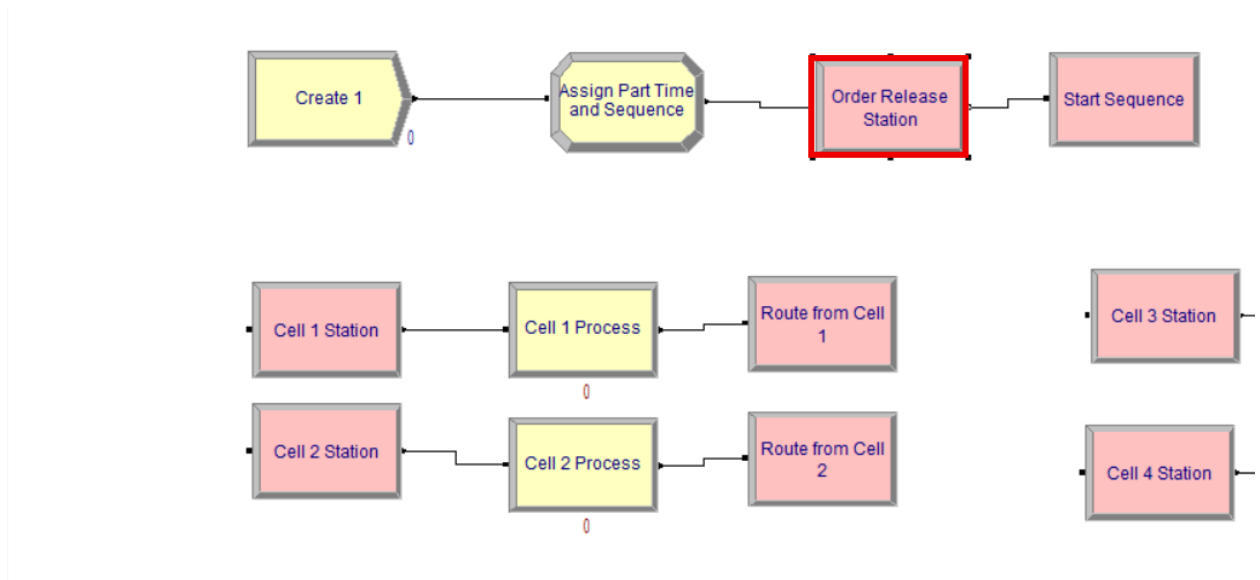
1. Create 1



2. Assign Part Time and Sequence

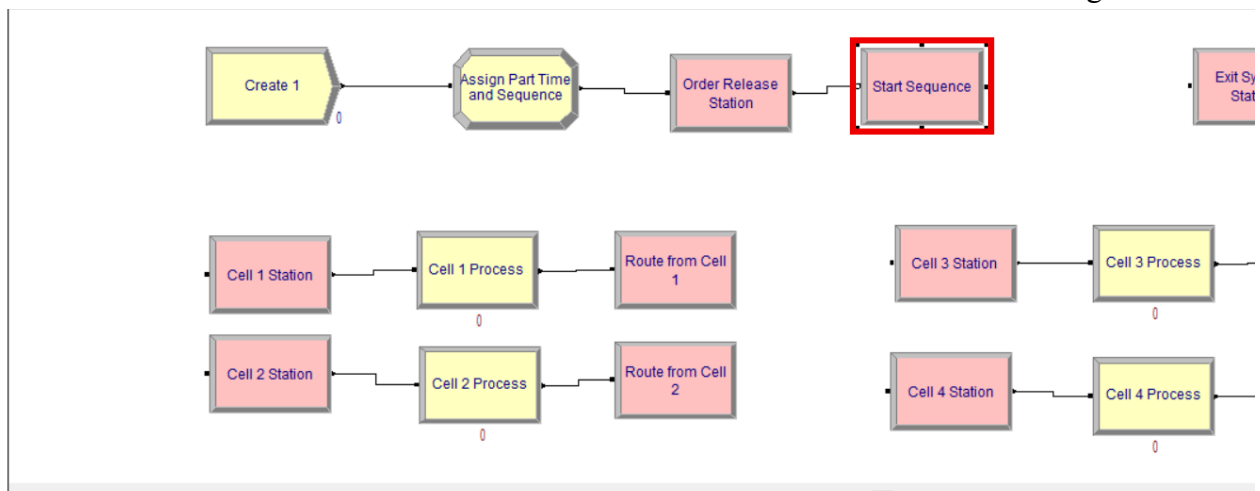


3. Stations : Same method for all Stations. One shown in Red-Box for better understanding.



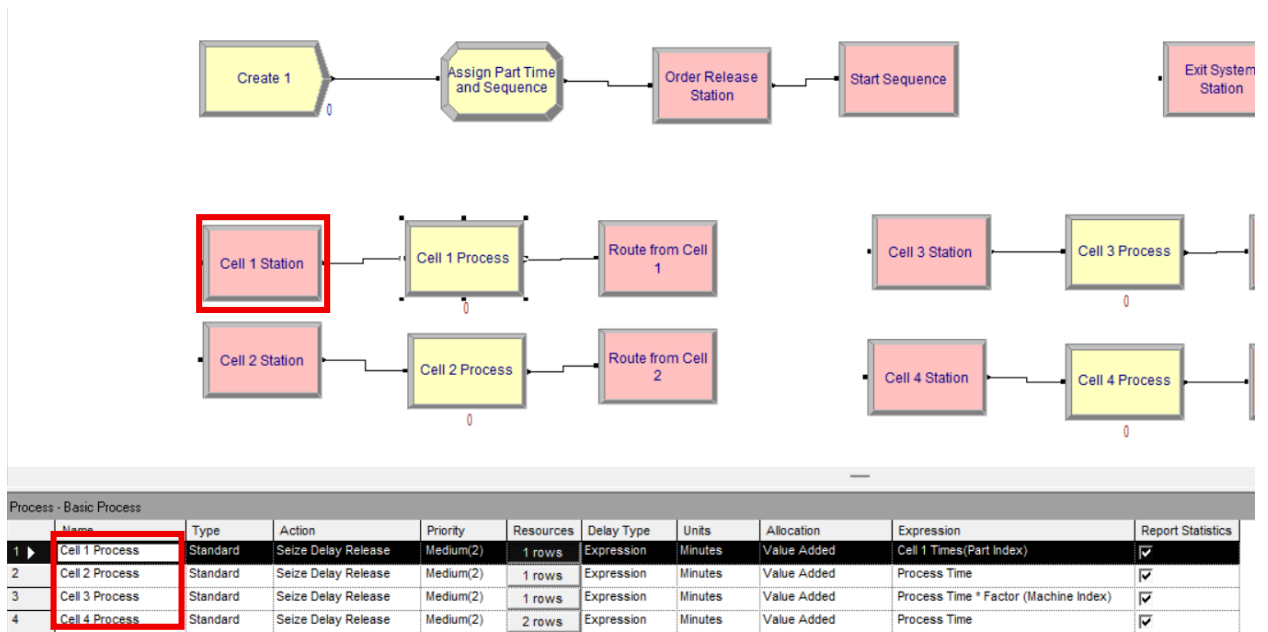
Station - Advanced Transfer						
	Name	Station Type	Station Name	Parent Activity Area	Associated Intersection	Report Statistics
1	Order Release Station	Station	Order Release Station			✓
2	Exit System Station	Station	Exit System Station			✓
3	Cell 1 Station	Station	Cell 1			✓
4	Cell 2 Station	Station	Cell 2			✓
5	Cell 3 Station	Station	Cell 3			✓
6	Cell 4 Station	Station	Cell 4			✓

4. Routes: Same method for all Routes. One shown in Red-Box for better understanding.



Route - Advanced Transfer				
	Name	Route Time	Units	Destination Type
1	Start Sequence	Transfer Time	Minutes	By Sequence
2	Route from Cell 1	Transfer Time	Minutes	By Sequence
3	Route from Cell 2	Transfer Time	Minutes	By Sequence
4	Route from Cell 3	Transfer Time	Minutes	By Sequence
5	Route from Cell 4	Transfer Time	Minutes	By Sequence

5. Process: Same method for all Process. One shown in Red-Box for better understanding.



6. Variables



Variable



Schedule



Set

Advanced Transfer

Advanced Process

Variable - Basic Process

	Name	Rows	Columns	Data Type	Clear Option	File Name	Initial Values	Report Statistics
1	Transfer Time			Real	System		1 rows	☐
2 ▶	Factor	2		Real	System		2 rows	☐

Double-click here to add a new row.

7. Entity

Entity - Basic Process									
	Entity Type	Initial Picture	Holding Cost / Hour	Initial VA Cost	Initial NVA Cost	Initial Waiting Cost	Initial Tran Cost	Initial Other Cost	Report Statistics
1	Entity 1	Picture.Report	0.0	0.0	0.0	0.0	0.0	0.0	☒
2	Part 1	Picture.Report	0.0	0.0	0.0	0.0	0.0	0.0	☒
3	Part 2	Picture.Report	0.0	0.0	0.0	0.0	0.0	0.0	☒
4	Part 3	Picture.Report	0.0	0.0	0.0	0.0	0.0	0.0	☒

Double-click here to add a new row.

8. Set

Set - Basic Process

	Name	Type	Members
1 ▶	Entity Types	Entity Type	3 rows
2	Part Pictures	Entity Picture	3 rows
3	Cell 3 Machines	Resource	2 rows

Double-click here to add a new row.

9. Resource

Resource - Basic Process

	Name	Type	Capacity	Busy / Hour	Idle / Hour	Per Use	StateSet Name	Failures	Report Statistics
1 ▶	Cell 3 New	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓
2	Cell 3 Old	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓
3	Cell 1 Machine	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓
4	Cell 2 Machine	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓
5	Cell 3 Machine	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓
6	Cell 4 Machines	Fixed Capacity	1	0.0	0.0	0.0		0 rows	✓

Double-click here to add a new row.

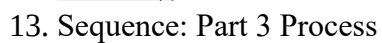
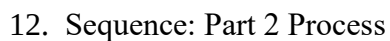
10. Queue

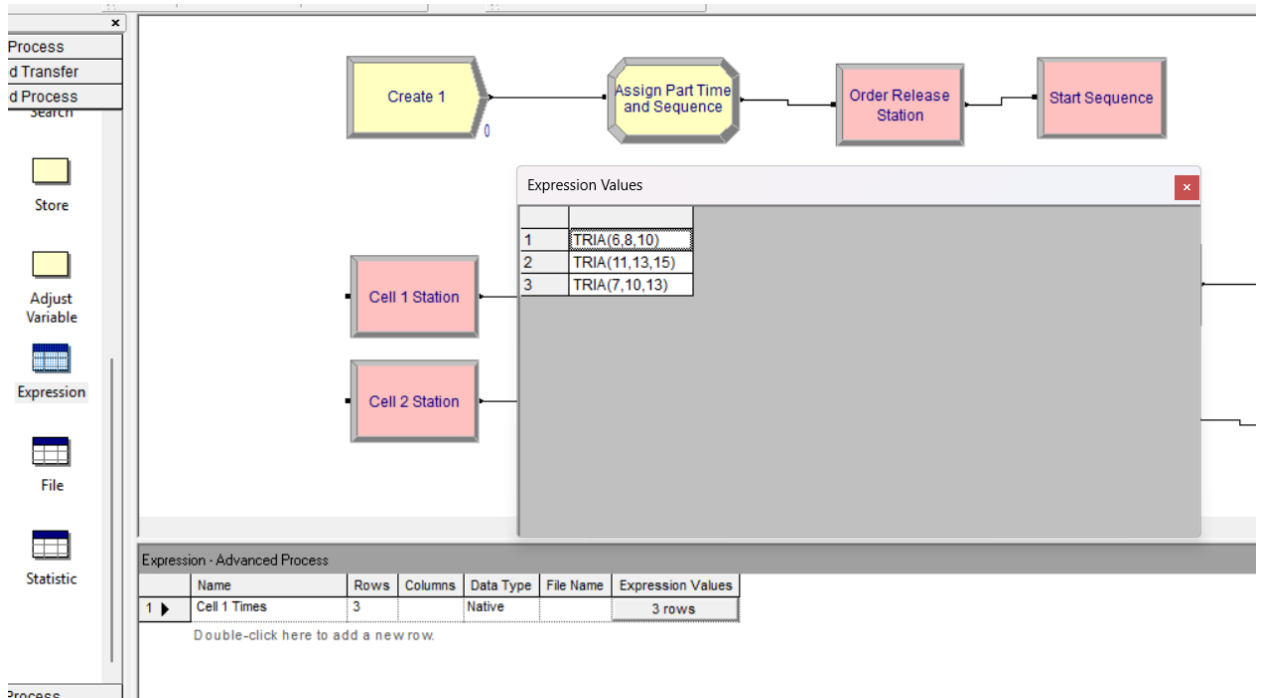
Queue - Basic Process

	Name	Type	Shared	Report Statistics
1 ▶	Cell 1 Process.Queue	First In First Out	☐	✓
2	Cell 2 Process.Queue	First In First Out	☐	✓
3	Cell 3 Process.Queue	First In First Out	☐	✓
4	Cell 4 Process.Queue	First In First Out	☐	✓

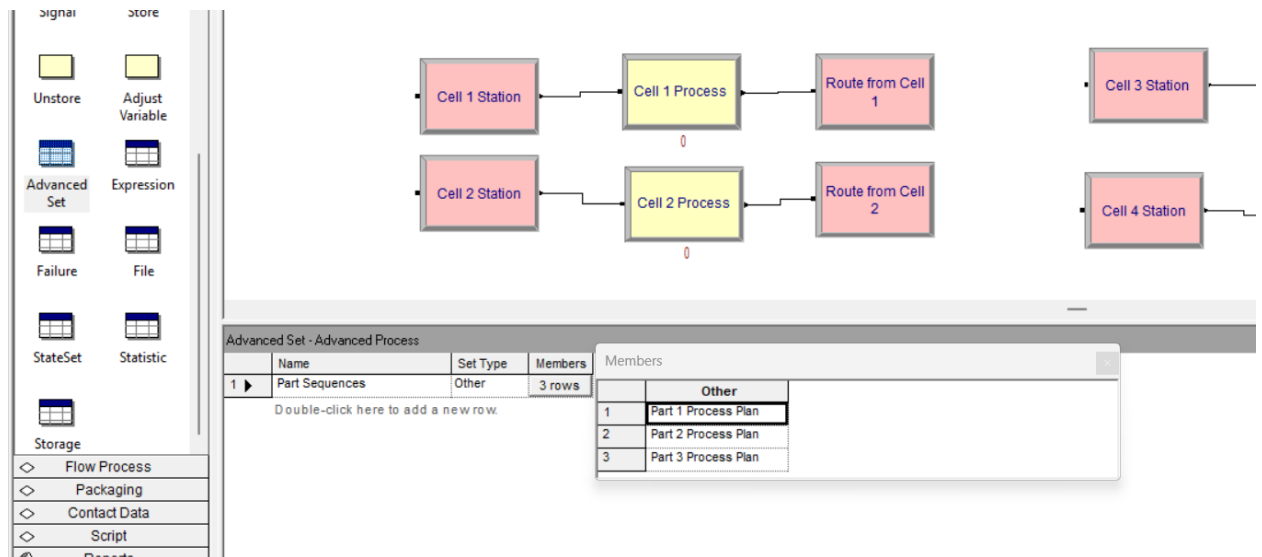
Double-click here to add a new row.

11. Sequence: Part 1 Process





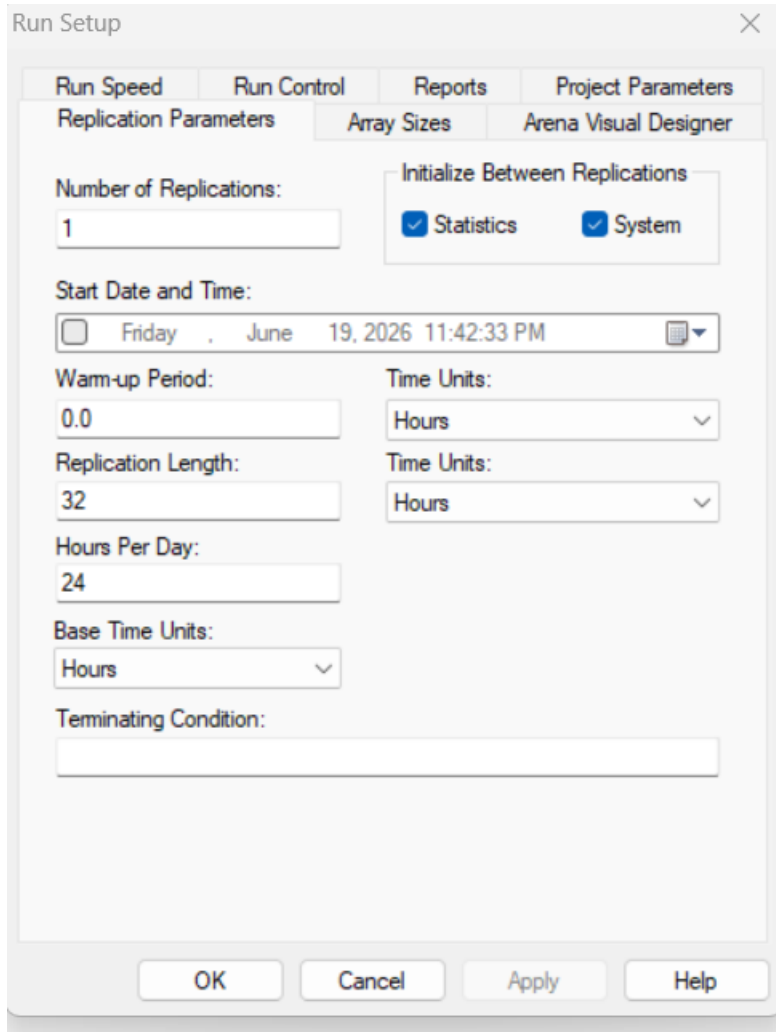
15. Advanced Set



16. Dispose 1

Dispose - Basic Process		
	Name	Record Entity Statistics
1 ▶	Dispose 1	☒

Run Setup

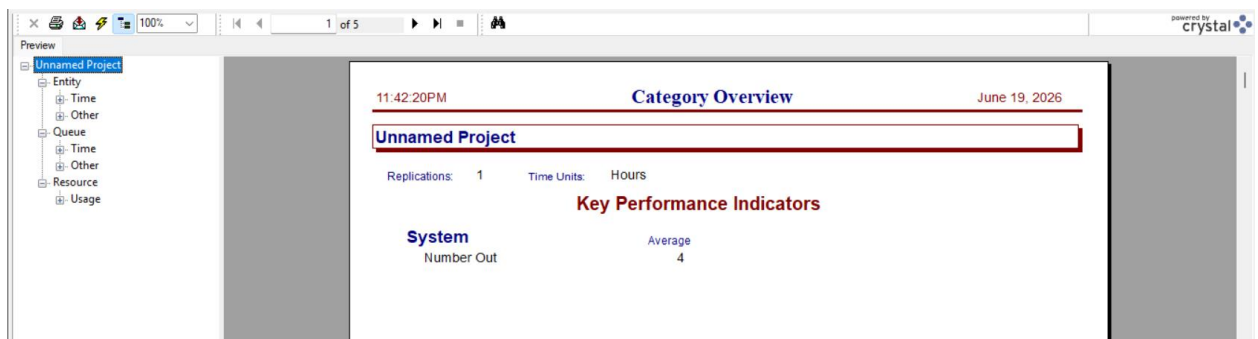


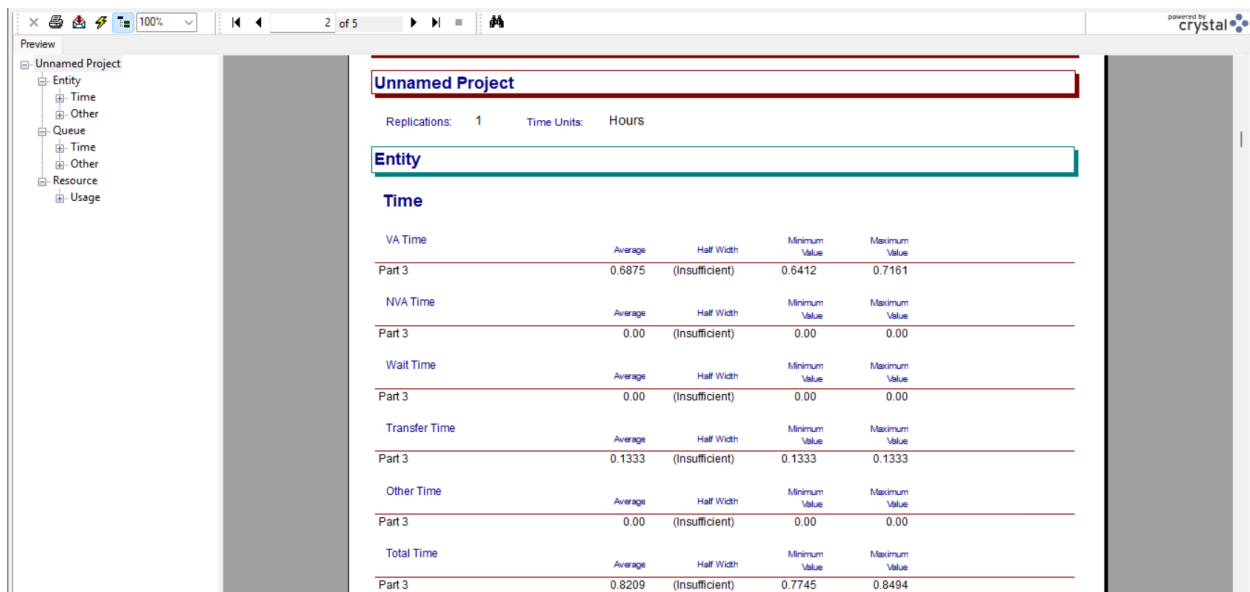
The Run Setup dialog box is shown with the following settings:

- Run Speed:** (tab selected)
- Run Control:** (tab selected)
- Reports:** (tab selected)
- Project Parameters:** (tab selected)
- Replication Parameters:** (sub-tab selected)
- Array Sizes:** (sub-tab selected)
- Arena Visual Designer:** (sub-tab selected)
- Number of Replications:** 1
- Initialize Between Replications:**
 - ☒ Statistics
 - ☒ System
- Start Date and Time:** Friday, June 19, 2026 11:42:33 PM
- Warm-up Period:** 0.0
- Time Units:** Hours
- Replication Length:** 32
- Time Units:** Hours
- Hours Per Day:** 24
- Base Time Units:** Hours
- Terminating Condition:** (empty text box)

Buttons: OK, Cancel, Apply, Help

Results

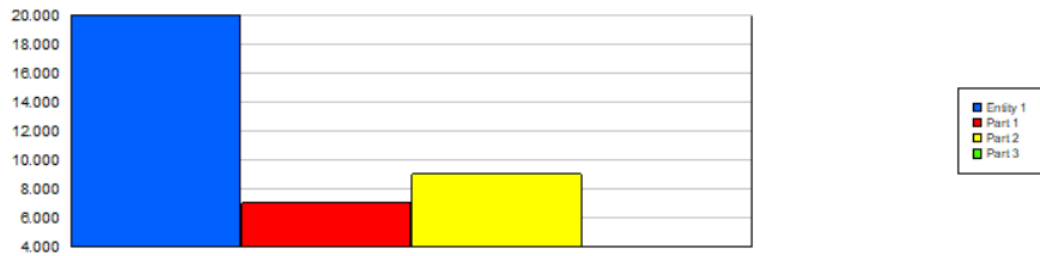


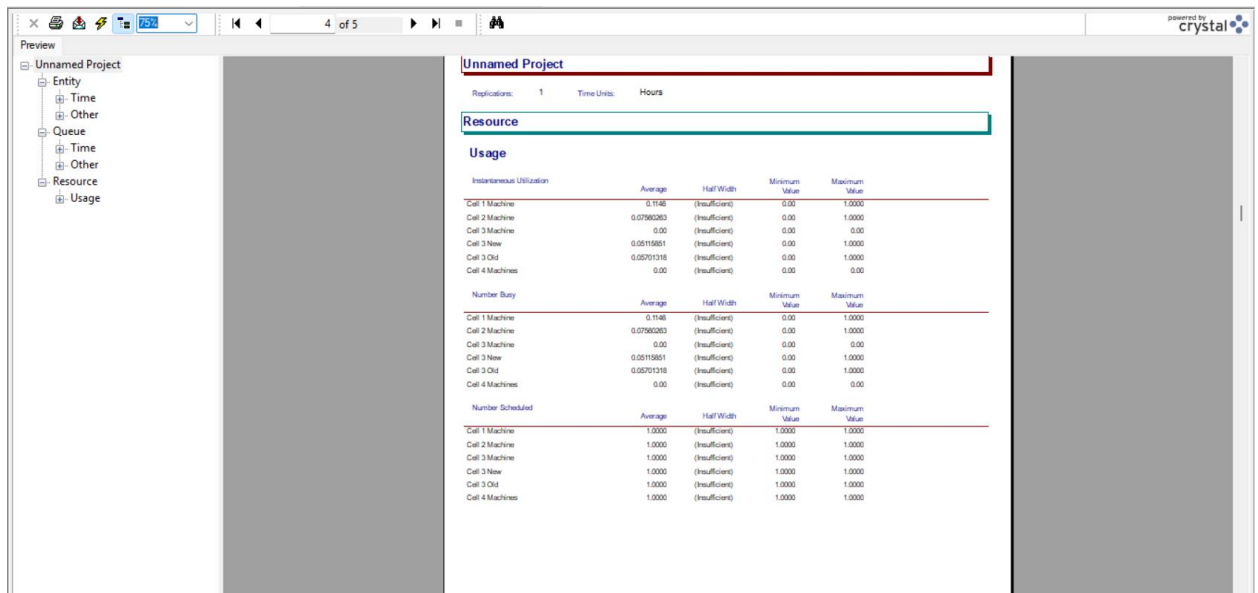
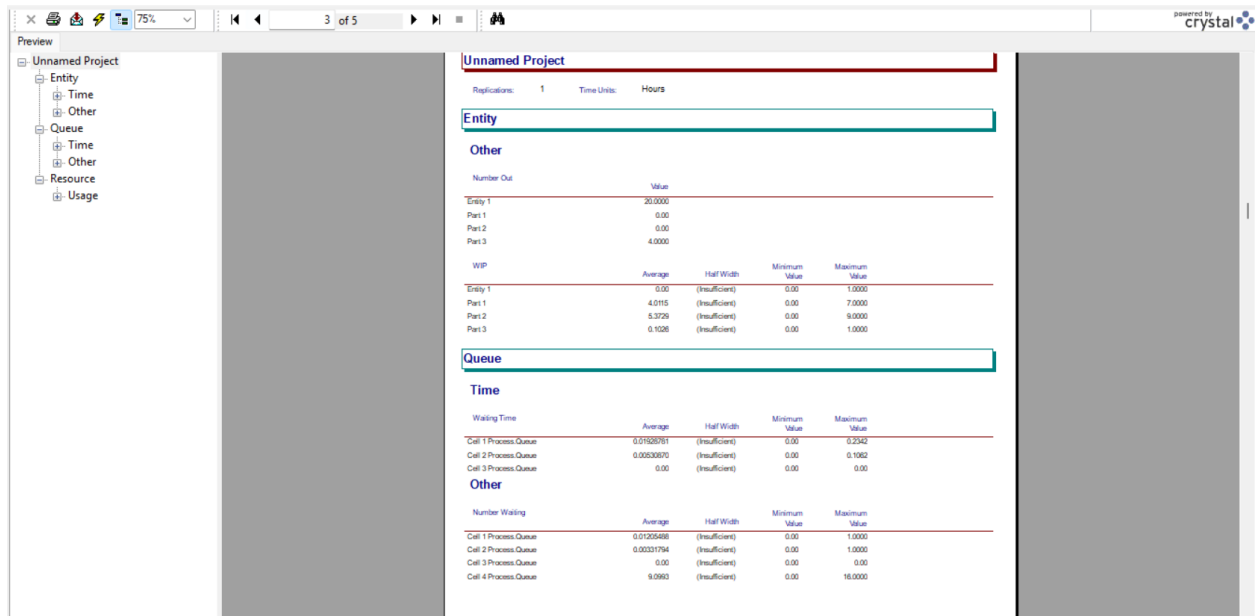


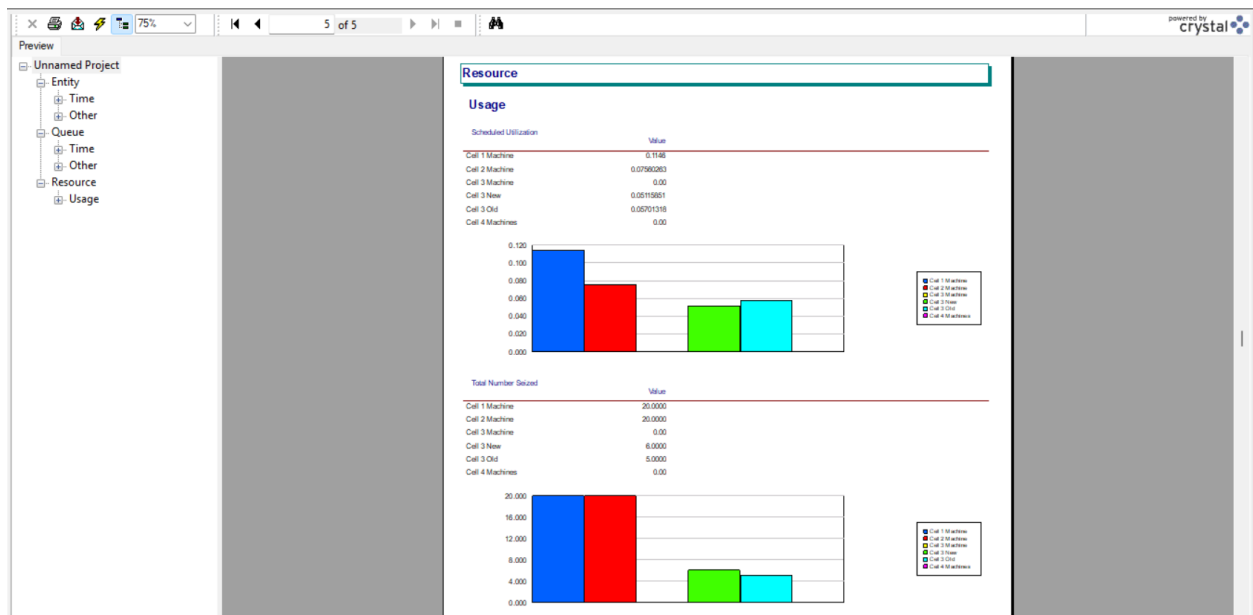
Other

Number In

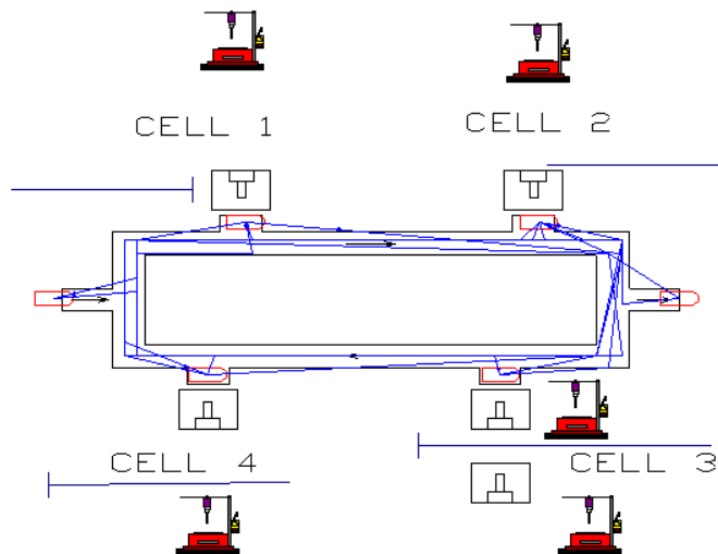
	Value
Entity 1	20.0000
Part 1	7.0000
Part 2	9.0000
Part 3	4.0000







Animation



Experiment 7: Simulation and Visualization of a Restaurant Capacity Planning Problem

Objectives

1. To model a real-life restaurant system in Arena by representing customer arrivals, table allocation, and waiting behavior under limited seating capacity.
2. To analyze customer arrival patterns and group classification, incorporating probabilistic arrivals of single customers, couples, and groups of three to reflect real service variability.
3. To evaluate capacity utilization and queue dynamics, including how limited resources (tables) affect waiting time, blocking, and system congestion.
4. To measure customer satisfaction indicators, such as average waiting time, probability of waiting, and system performance under different demand scenarios.

Software

- Rockwell Arena Simulation/AnyLogic

Introduction

Service sector capacity planning is the process of determining and managing the resources required to meet customer demand efficiently while maintaining the desired level of service quality. In service systems such as restaurants, banks, hospitals, and call centers, capacity planning involves decisions regarding the number of service facilities, employees, and schedules needed to handle varying customer arrivals. Intermittent scheduling refers to the allocation of resources in environments where demand fluctuates over time and service requests do not occur continuously at a constant rate. Simulation modeling provides a powerful tool for analyzing such systems because it allows decision-makers to evaluate different capacities and scheduling policies without disrupting real operations.

In a restaurant environment, customers may arrive individually or in groups of different sizes, creating variability in demand and resource utilization. Simulation software such as **Arena** enables the modeling of customer group behavior, including arrival patterns, waiting decisions, service times, and table occupancy. Time-persistent schedules can be incorporated to represent changes in staffing levels or customer arrival rates throughout the day. Human resource pooling allows employees to serve multiple tasks or customer groups, improving operational flexibility and resource utilization. Performance measures such as average waiting time, queue length, table utilization, throughput, and customer satisfaction metrics help evaluate the effectiveness of the restaurant's capacity plan. By visualizing system behavior and analyzing simulation results,

managers can identify bottlenecks, improve scheduling decisions, and enhance overall service quality while minimizing customer waiting and resource idle time.

Problem Statement

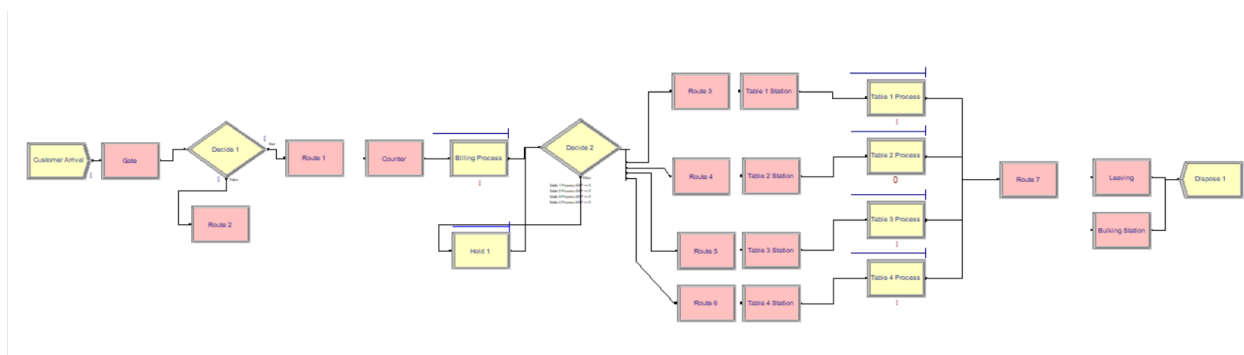
A restaurant has 4 tables available for serving customers. Customer groups arrive randomly throughout the day according to the following probability distribution:

- Couples (2 persons): 70%
- Single customers (1 person): 20%
- Groups of 3 people: 10%

Each arriving group requires one table and occupies it for a certain dining duration. When a group arrives and at least one table is available, it is immediately assigned a table and begins dining. However, if all four tables are occupied, the arriving group joins a waiting queue and waits until a table becomes available. Customers are served on a first-come, first-served (FCFS) basis.

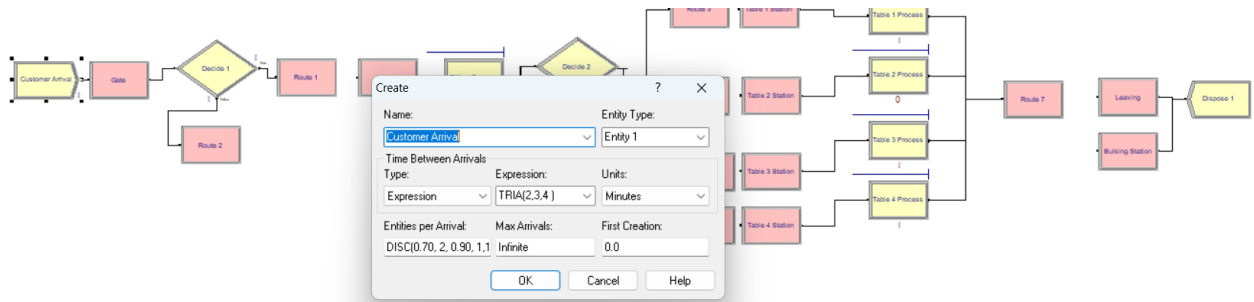
Using Arena Simulation Software, develop a simulation model of the restaurant system to represent customer arrivals, table allocation, dining service, and waiting behavior. Run the simulation for a specified operating period and analyze key performance indicators such as average customer waiting time, average queue length, table utilization, number of customers served, and the percentage of customers who experience waiting. Based on the simulation results, evaluate the effectiveness of the restaurant's current capacity and identify opportunities for improving customer service and resource utilization.

Model

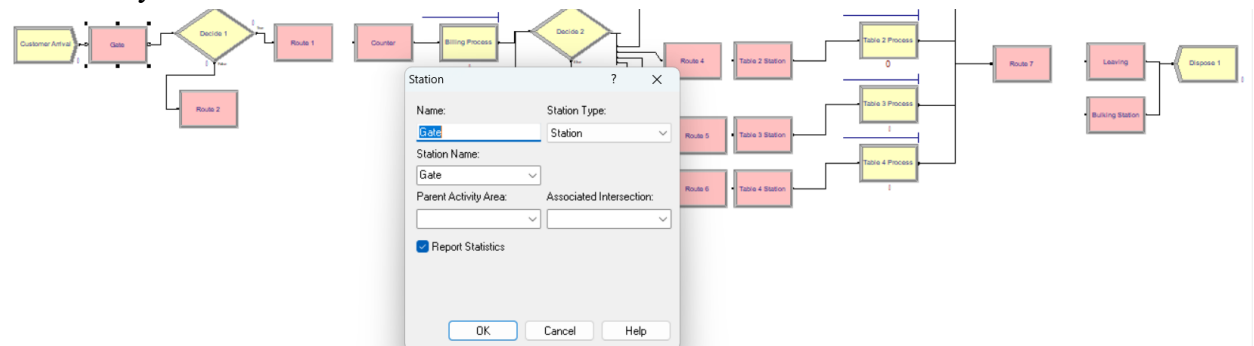


Configurations

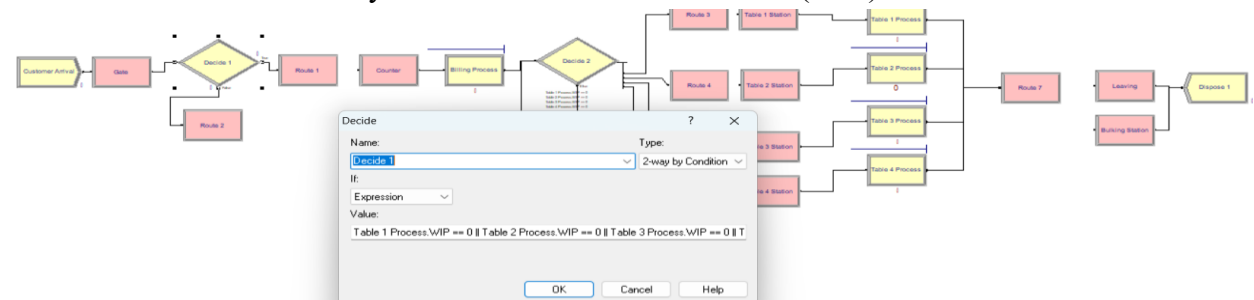
1. Customer Arrival : Customer arrives at the restaurant.



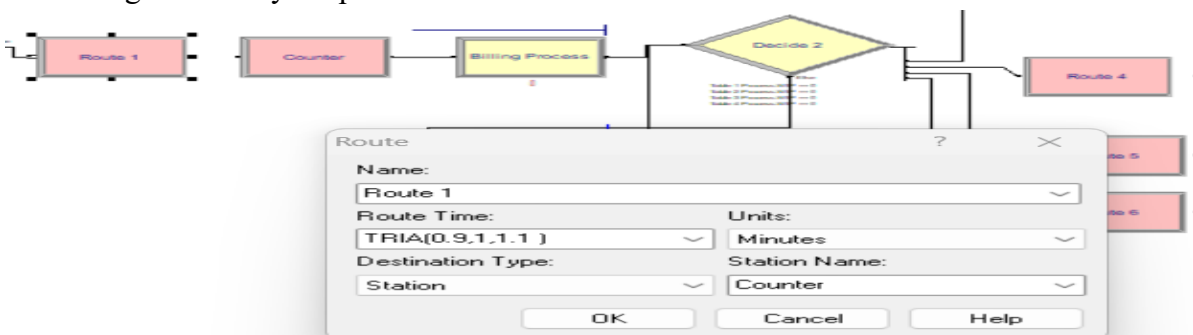
2. Gate: Entry to restaurant.



3. Decide 1 : Customer sits only if there is no Work -in- Process (WIP)



4. Route 1: If tables are empty then proceeds to counter. Otherwise proceed to Route 2 then to Bulking and finally Dispose.



5. Billing Process

Process

Name: Billing Process Type: Standard

Logic

Action: Seize Delay Release Priority: Medium(2)

Resources:

- Set, Billing Counter, 1, Cyclical, <End of list>

Buttons: Add..., Edit..., Delete

Delay Type: Expression Units: Minutes Allocation: Value Added

Expression: TRIA(1.5,2,2.5)

☒ Report Statistics

Buttons: OK, Cancel, Help

6. Decide 2: Customer decides in which table to sit.

Decide

Name: Decide 2 Type: N-way by Condition

Conditions:

- Expression, Table 1 Process.WIP == 0
- Expression, Table 2 Process.WIP == 0
- Expression, Table 3 Process.WIP == 0
- Expression, Table 4 Process.WIP == 0
- <End of list>

Buttons: Add..., Edit..., Delete

Buttons: OK, Cancel, Help

7. Hold 1: If there in empty table then Hold (wait).

Hold

Name: Hold 1 Type: Scan for Condition

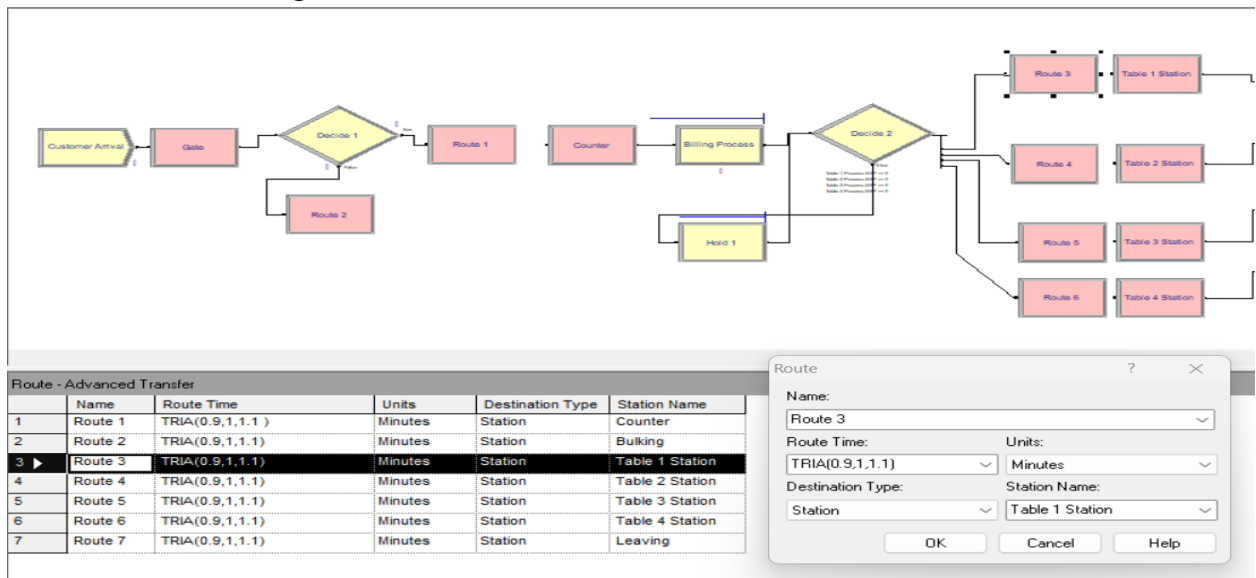
Condition: Table 1 Process.WIP == 0 || Table 2 Process.WIP == 0 || Ta

Queue Type: Queue

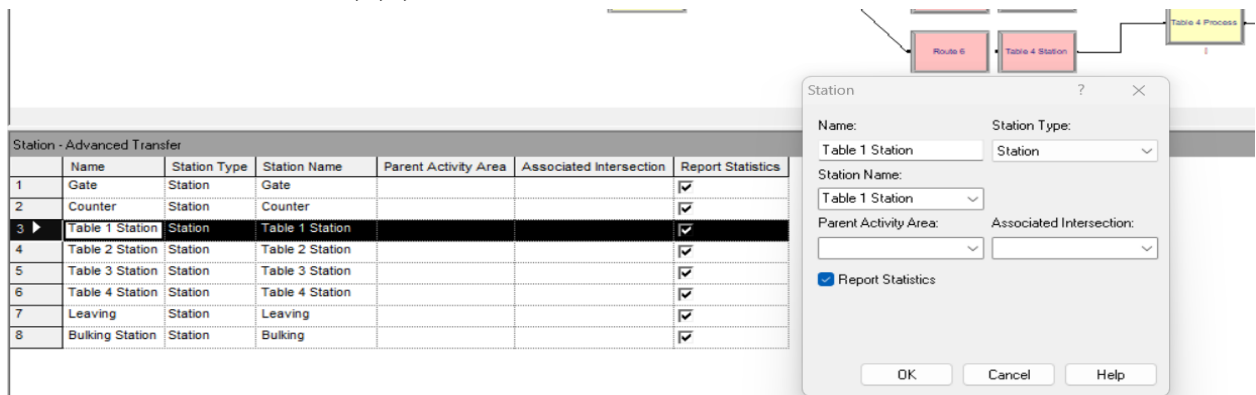
Queue Name: Hold 1.Queue

Buttons: OK, Cancel, Help

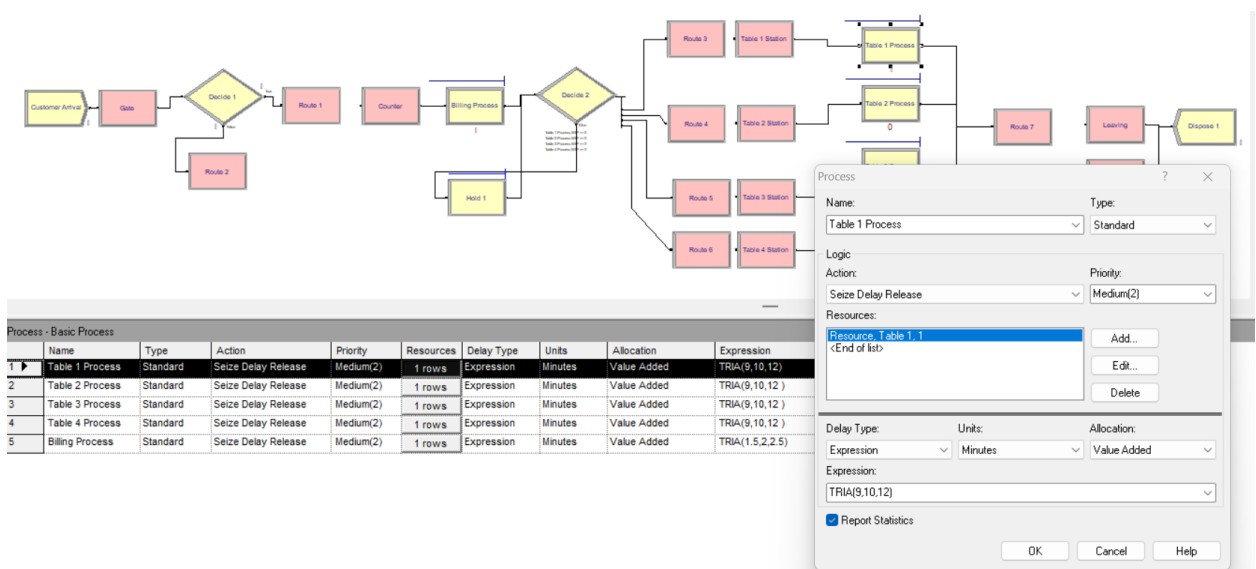
8. Route 3,4,5,6 : Routing customers to tables.



9. Station: Stations of Table 1,2,3,4.



10. Process: Processes of the tables.



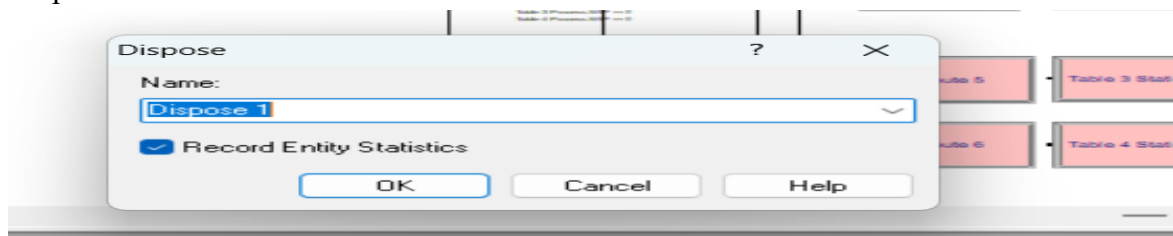
11. Route 7: Routing customers to leave after meals.



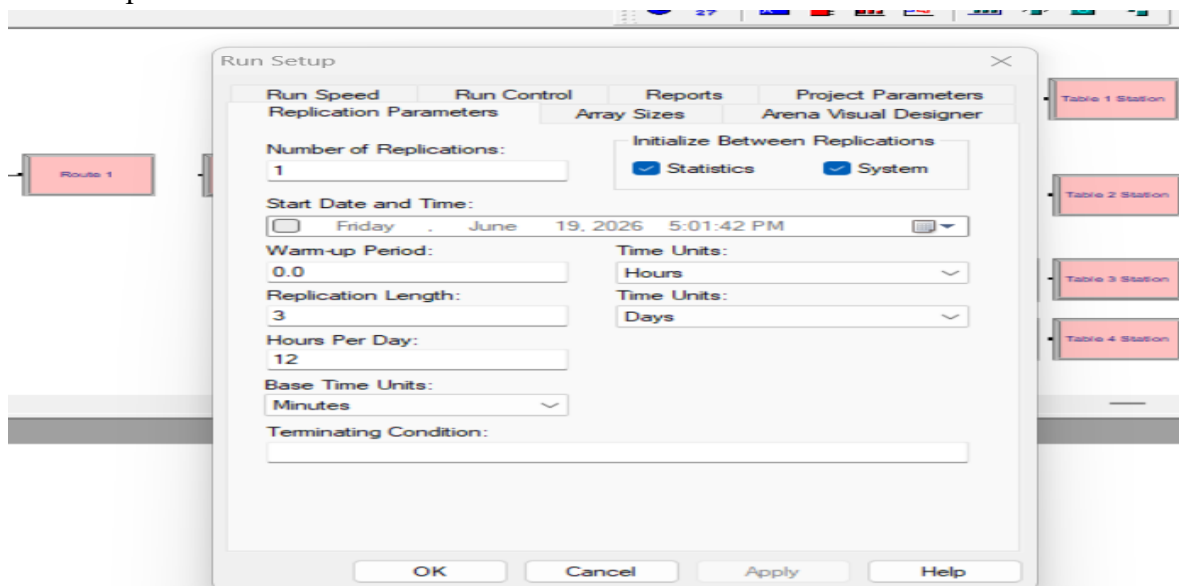
12. Leaving and Bulking

7	Leaving	Station	Leaving			☒
8 ▶	Bulking Station	Station	Bulking			☒

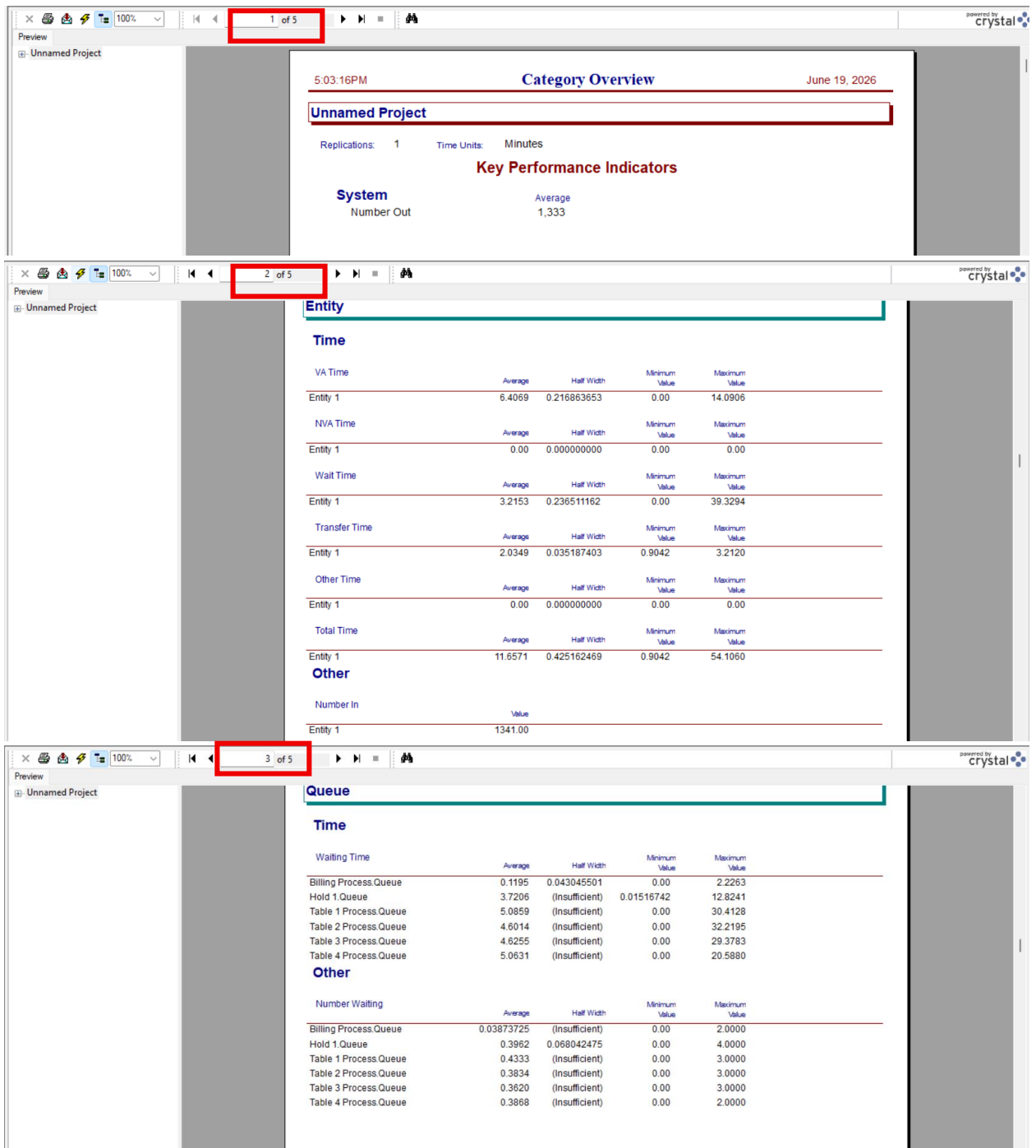
13. Dispose 1: End of simulation.

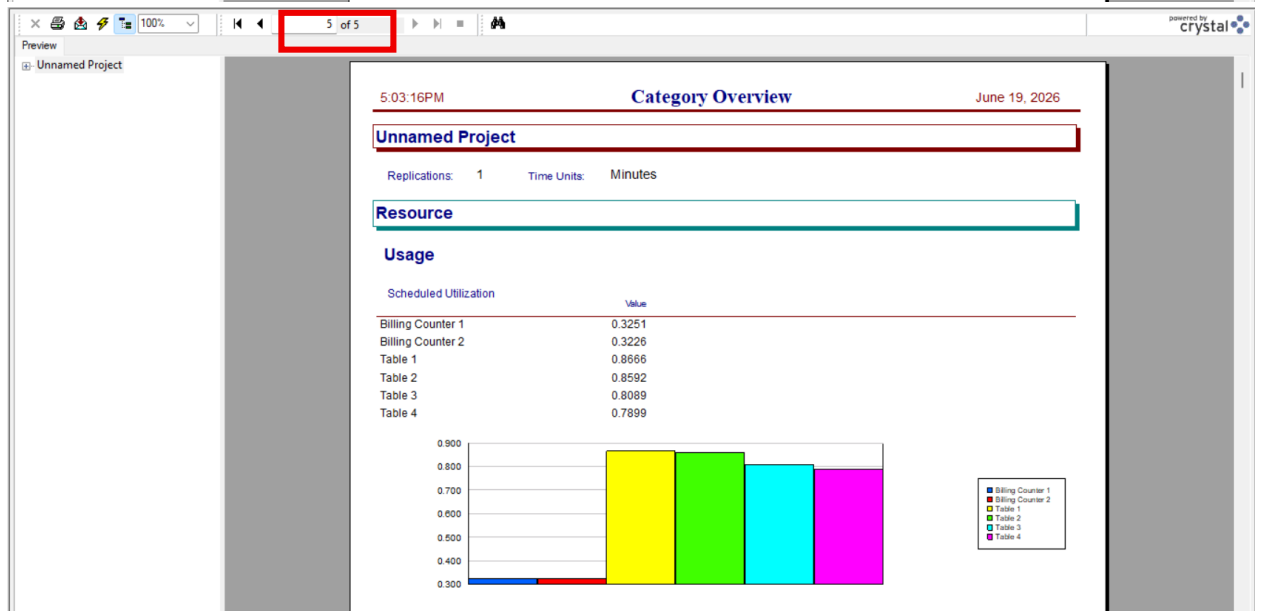
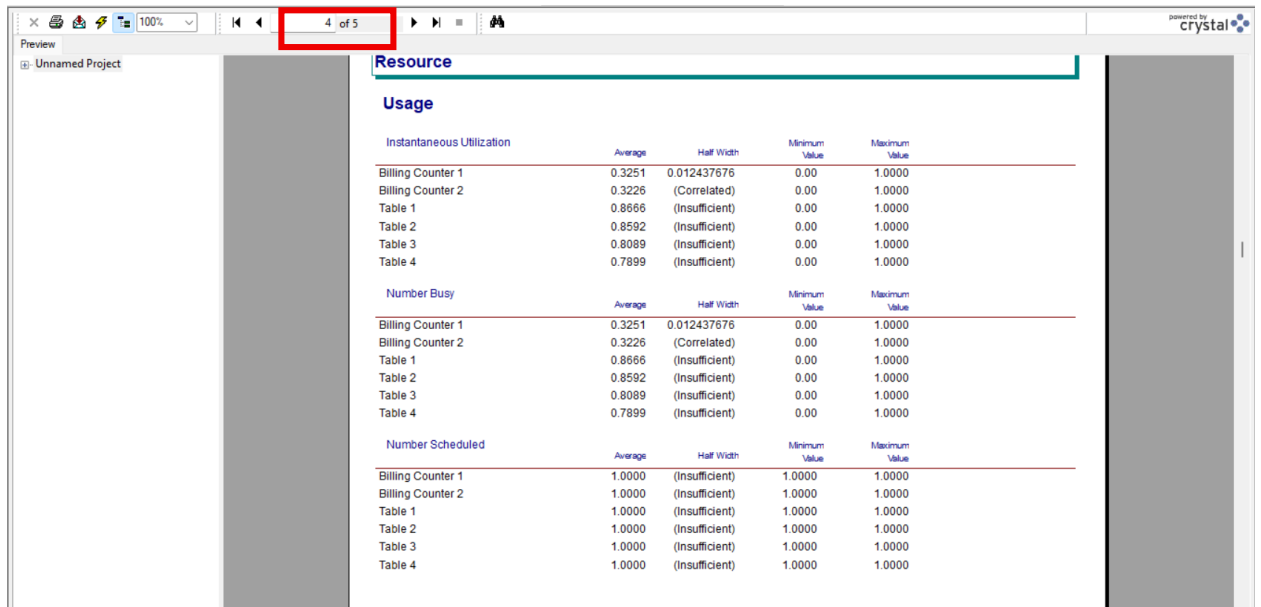


14. Run Setup



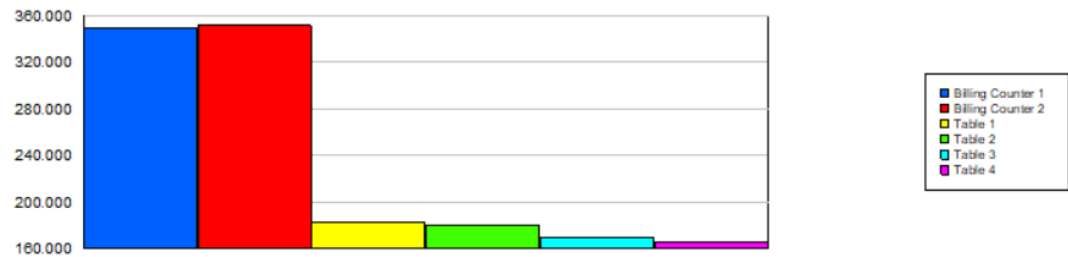
15. Results:





Total Number Seized

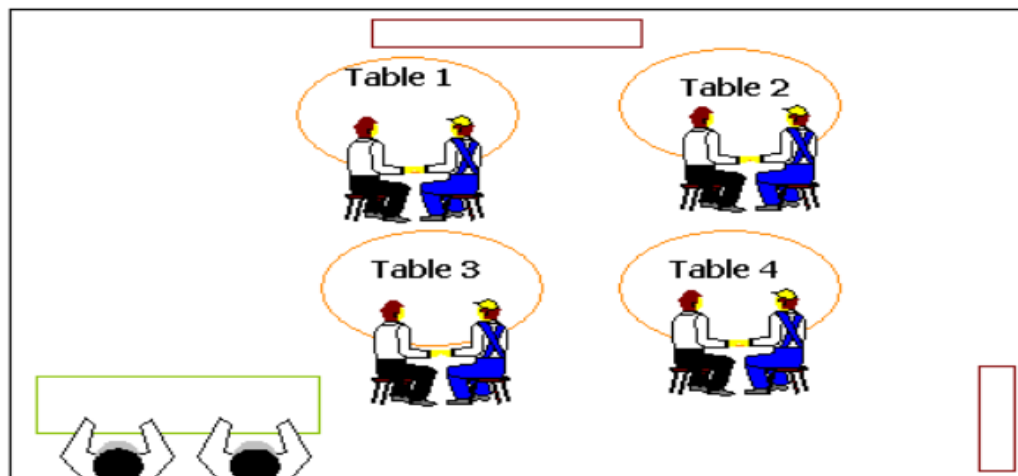
	Value
Billing Counter 1	349.00
Billing Counter 2	351.00
Table 1	182.00
Table 2	180.00
Table 3	169.00
Table 4	165.00



Model Filename: C:\Users\SHABIB\Desktop\ArenaModel2_restaurant

Page 5 of 5

Animation



Experiment 8: Simulation and visualization of bus scheduling and parking lot systems.

Objective

The objective of this experiment is to:

- Construct and execute a simulation model that replicates a campus transit terminal handling both standard shuttles and articulated buses.
- Assess Queue Dynamics and Traffic Delays for both vehicle types to measure and compare the average waiting times and maximum queue lengths
- Quantify Resource Utilization and identify facility Bottlenecks to evaluate the utilization rates of the single-capacity parking bays.

Software

- Rockwell Arena Simulation/AnyLogic

Introduction

In urban planning and transit engineering, terminal design relies heavily on balancing infrastructure capacity with variable arrival patterns. Transit hubs frequently act as operational bottlenecks because they must accommodate diverse vehicle fleets—ranging from standard fleet shuttles to high-capacity articulated units—each requiring different physical footprints and processing times. If terminal bays are restricted to specific vehicle types, a mismatch between fleet distribution and bay allocation can quickly trigger systemic delays.

Managing these facilities requires a deep understanding of resource utilization and queue behavior. When a service bay is occupied, incoming vehicles are forced into physical queues, consuming spatial capacity and causing upstream traffic backlogs. Analyzing these dynamics helps determine whether a terminal's constraints stem from a lack of physical space, high service times, or an unequal distribution of fleet arrivals.

The objective of this study is to evaluate how localized spatial and resource constraints within a multi-bay transit facility impact overall system throughput. By examining the relationship between vehicle arrival patterns, distinct service channel capacities, and queue build-up, this report isolates how specialized infrastructure restrictions create operational bottlenecks. Ultimately, this analysis provides the framework needed to optimize bay configuration and minimize vehicle idle times during peak operations.

Problem

A university campus operates a shuttle system to handle the morning student rush, where standard shuttle buses arrive at the central transit hub approximately every 3 minutes on average, following a random exponential arrival pattern. However, the system must now accommodate two different vehicle types: 75% of the incoming vehicles are standard campus shuttles, while the remaining 25% are massive articulated regional buses. The transit hub is equipped with a designated drop-off station featuring exactly two parking slots, but because of their physical size, Slot 1 is a large bay that can fit any vehicle, while Slot 2 is a restricted small bay that can only accommodate standard shuttles. Standard shuttles take a uniform distribution of 3 to 6 minutes to unload, whereas the larger articulated buses carry double the passenger volume and require a uniform distribution of 6 to 10 minutes to fully clear out. If an articulated bus arrives while the large bay is occupied, it cannot park in the small bay and is forced to idle in a backup line on the street, even if the small bay is completely empty. The objective of this simulation exercise is to model a 120-minute peak period to calculate the average traffic backup time for both vehicle types and determine how frequently the size restriction on Slot 2 causes the entire system to bottleneck.

Model

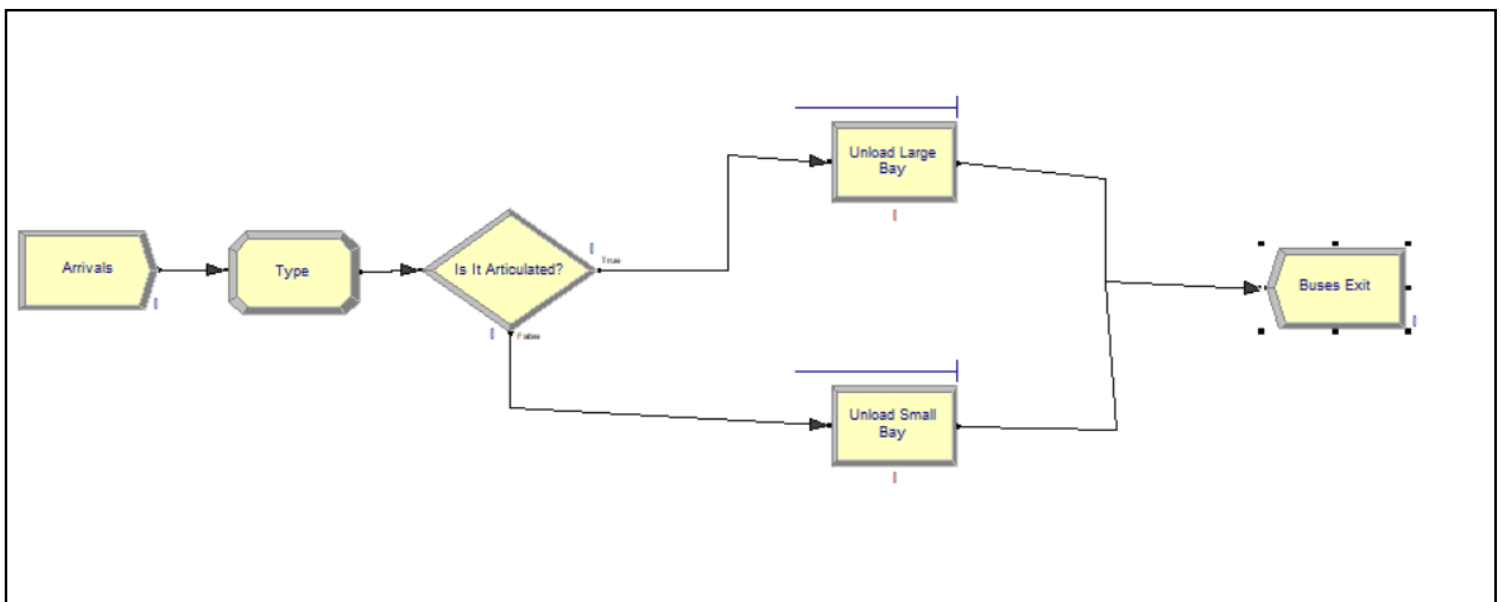
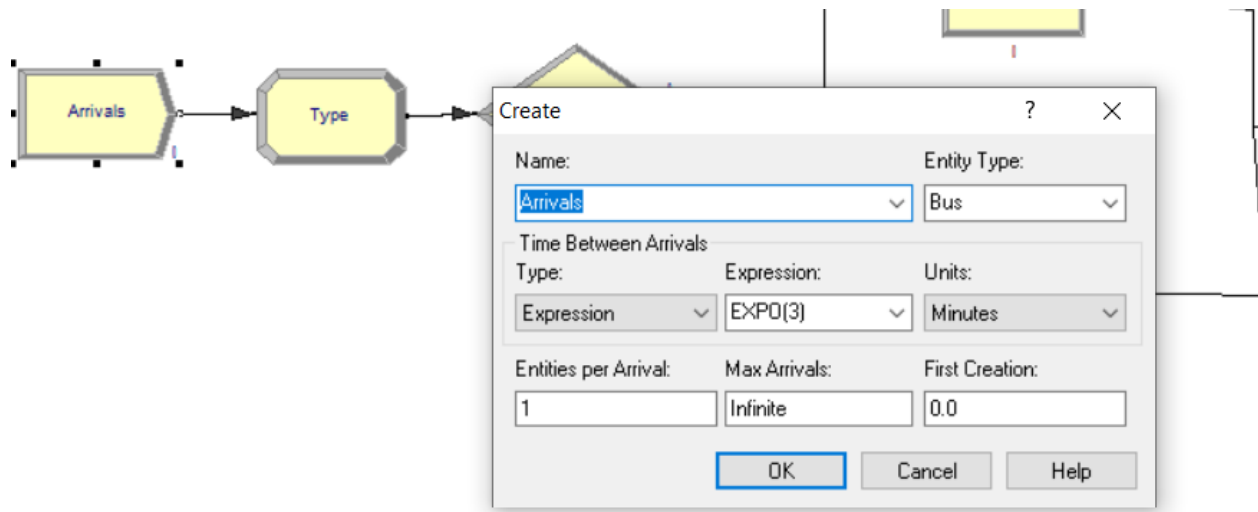


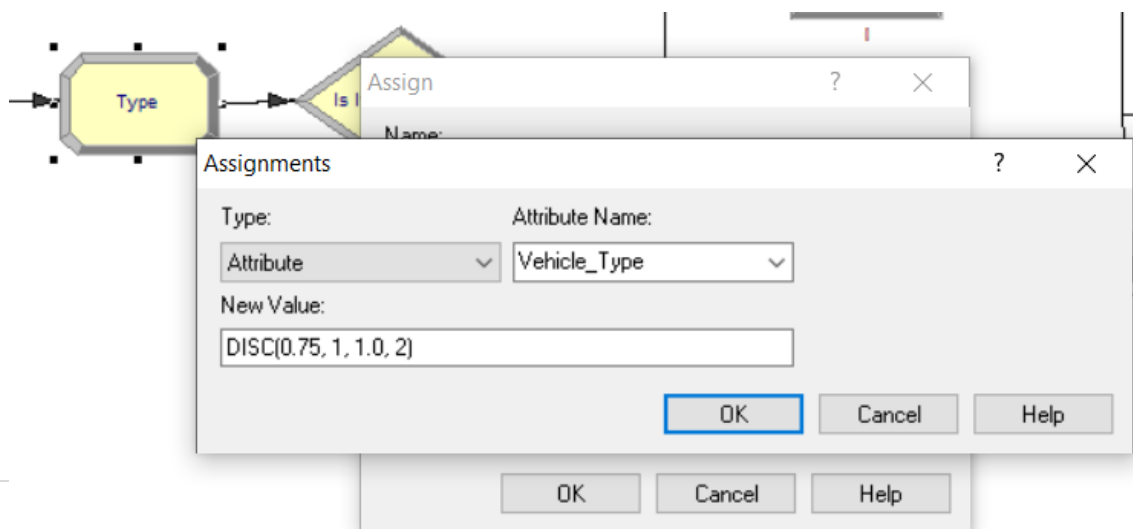
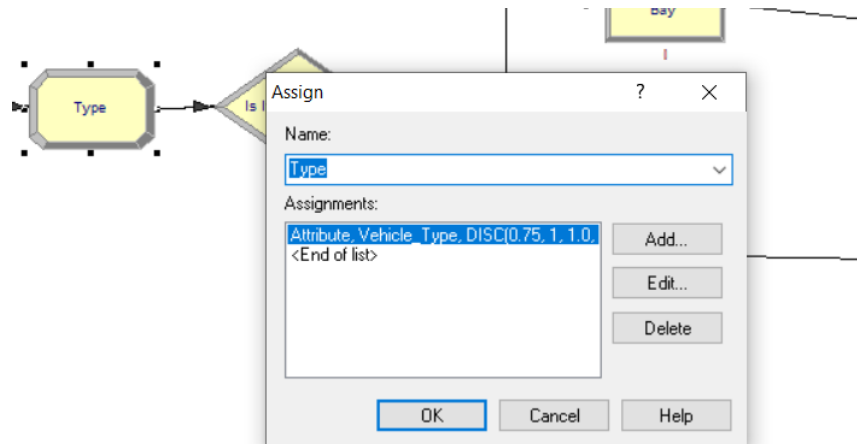
Figure 8.1: Simulation model of a parking lot system

Configurations

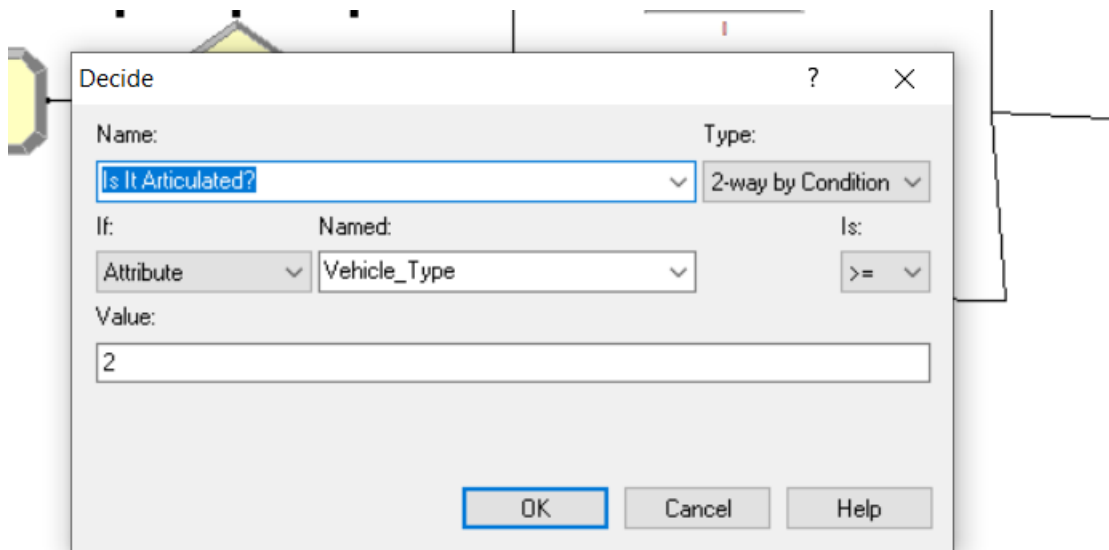
1. Arrival



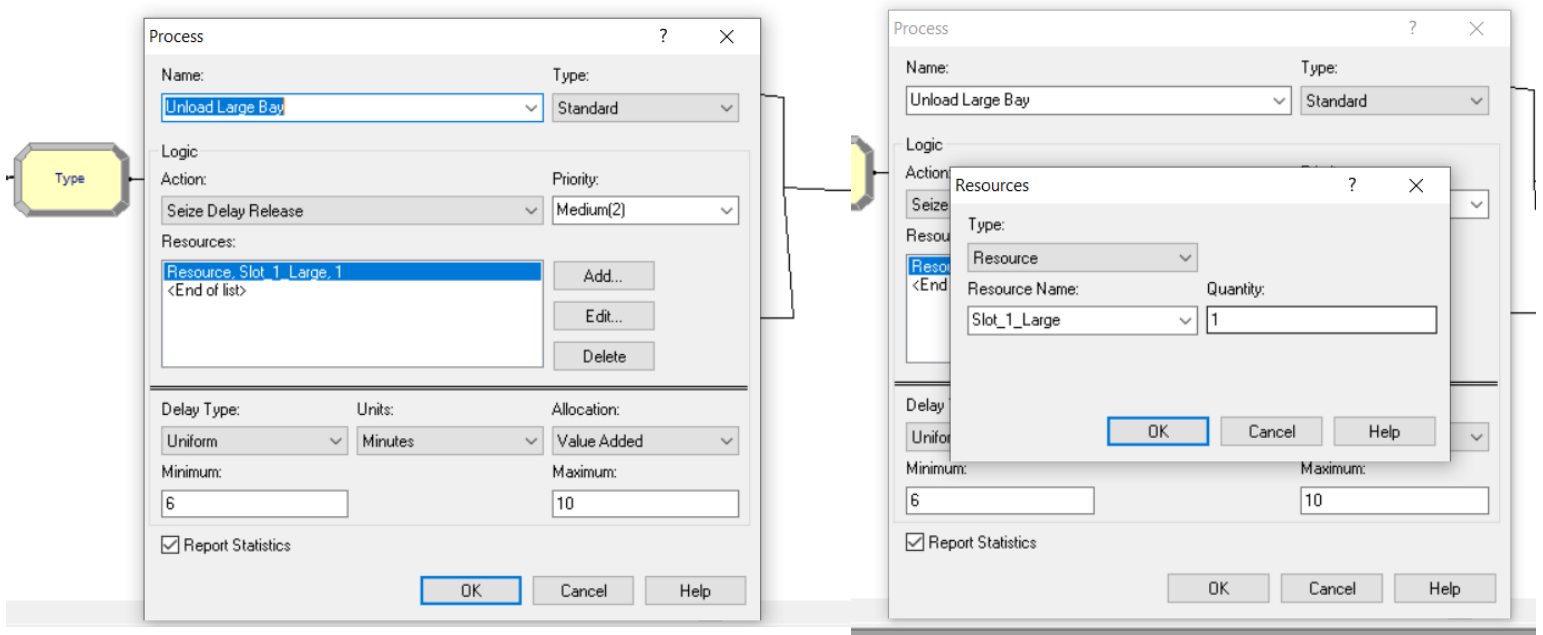
2. Type



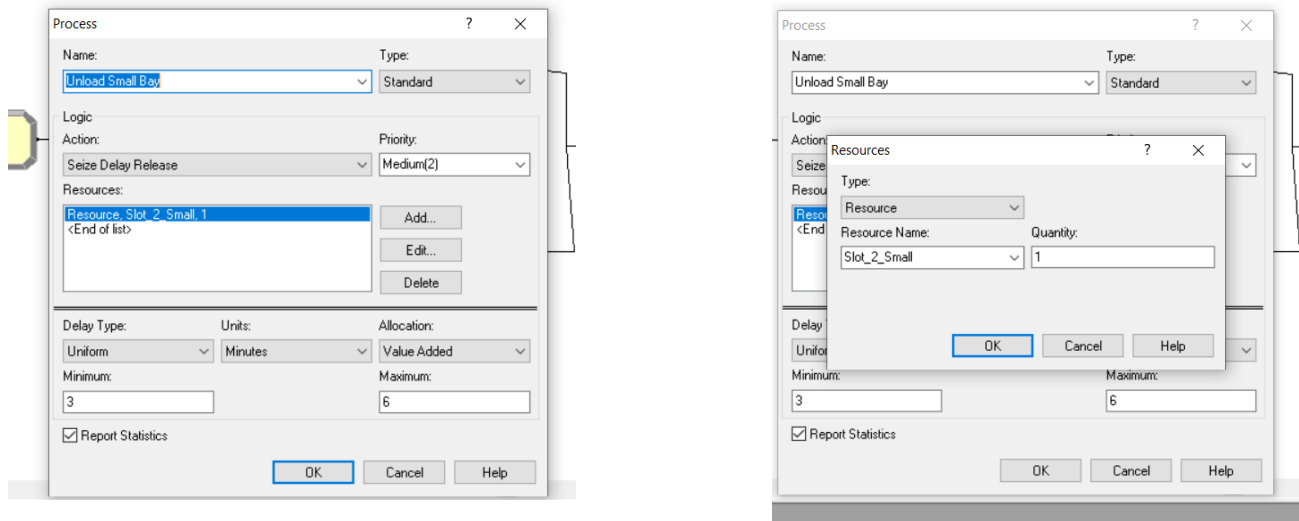
3. Is It Articulated?



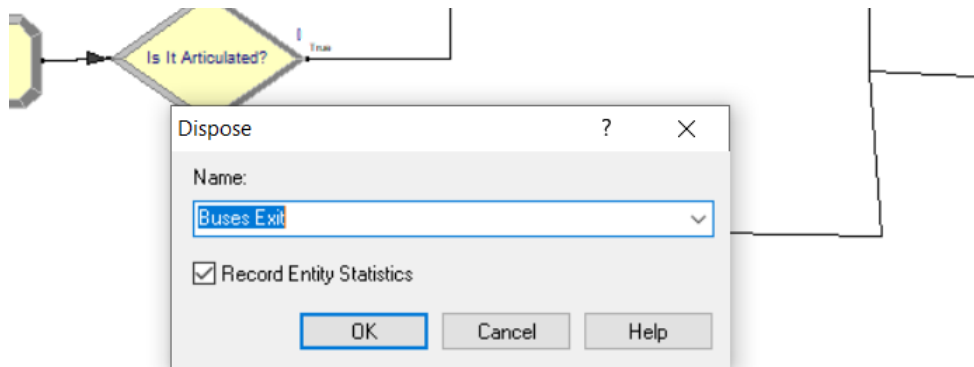
4. Unload Large Bay



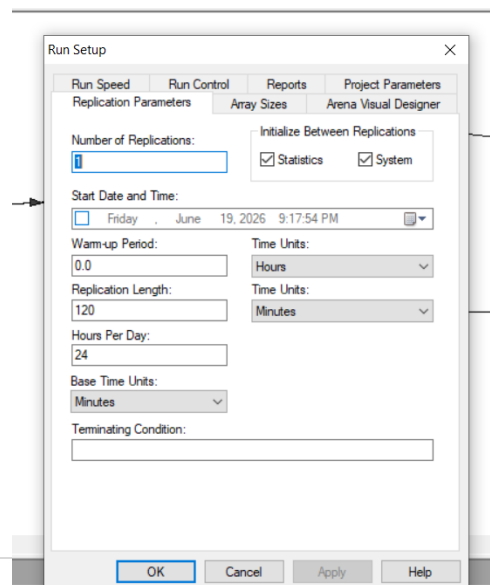
5. Unload Small Bay



6. Buses Exit



Run Set-Up



Experiment 9: Simulation and visualization of a traffic system.

Objective

The objective of this experiment is to:

- Design and implement a functional representation of a single-lane vehicle intersection using Arena Simulation Software Module
- Construct an independent control loop that automatically alternates signals to mimic a physical traffic light controller cycle.
- Collect and evaluate key performance indicators (KPIs) including average vehicle wait times, total throughput, and maximum queue lengths under a fixed traffic signal timing plan.

Software

- Rockwell Arena Simulation/AnyLogic

Introduction

Discrete-Event Simulation (DES) is a powerful mathematical modeling method where the operation of a system is represented as a chronological sequence of distinct, separate events. Unlike continuous simulations where variables change smoothly every millisecond, a DES model jumps instantly from one event time to the next (e.g., from a vehicle arriving, to a light changing, to a vehicle exiting).

In transportation engineering, DES is the industry standard for analyzing traffic networks. It allows engineers to model individual vehicles as unique entities moving through a system of resource constraints (like lanes and intersections) without the real-world risks, costs, or logistical disruptions of physical field testing.

To simulate a traffic intersection with an automated signal light, the simulation model must decouple the physical system from the logical control system. This requires two distinct parallel layers operating simultaneously within the model:

- The Physical Subsystem (Process Flow): This layer handles the physical entities (vehicles) entering the model boundaries. It controls vehicle generation, enforces the physical queue lines where cars wait during a red light, manages the limited spatial capacity of the crossing grid, and disposes of entities as they exit the system footprint.
- The Logical Subsystem (Control Loop): This layer acts as the background "brain" or hardware clock of the system. It uses an independent, invisible control entity trapped in a continuous, timed loop. As this control entity transitions through fixed delay blocks, it broadcasts system-wide signals that change the operational state of the physical subsystem (switching the lights between green and red).

Problem

Vehicles arrive at a single-lane approach to an intersection following an exponential distribution with a mean interarrival time of 6 seconds.

The intersection is regulated by a fixed-time traffic light controller. The signal control cycle operates on a repeating loop with the following parameters:

- Phase 1 (North-South): The light is Green for a constant duration of 30 seconds.
- Phase 2 (East-West): The light switches, allowing opposing traffic to pass for a constant duration of 30 seconds.

When the light is Red, arriving vehicles must come to a stop and queue dynamically at the stop line. When the light turns Green, vehicles are released from the hold queue and enter the intersection zone. It takes a constant traversal time of 2 seconds for a single vehicle to safely clear the intersection bottleneck. The intersection operates under a strict capacity limit, allowing only one vehicle to occupy the crossing zone at a time.

Design, animate, and execute a discrete-event simulation model in Rockwell Arena to evaluate system performance over an 8-hour operational shift. The primary objective is to analyze the relationship between the fixed 30-second signal cycle and the resulting traffic metrics, specifically monitoring total vehicle throughput, maximum queue accumulation, and average driver wait times.

Model

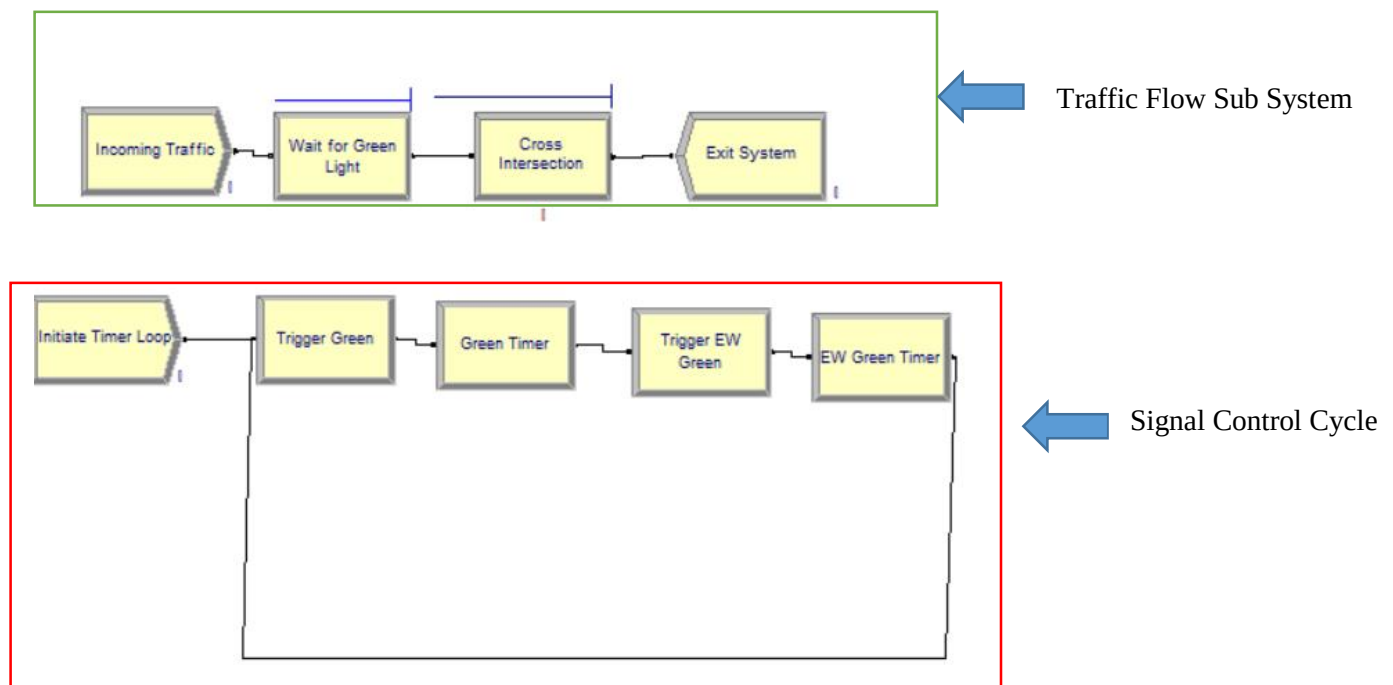
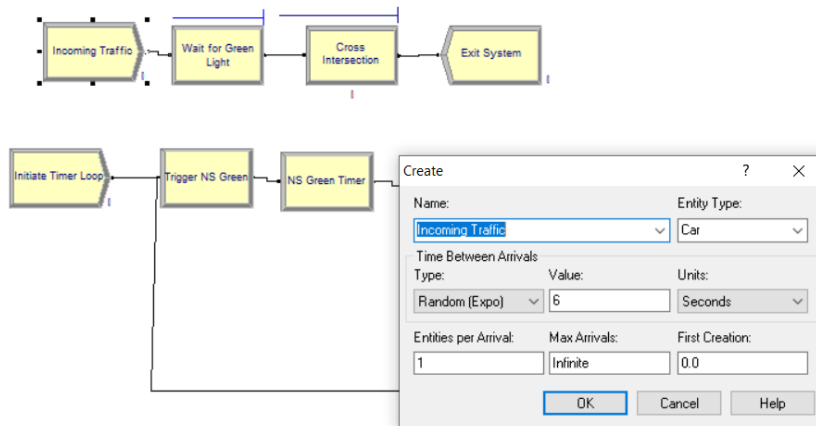


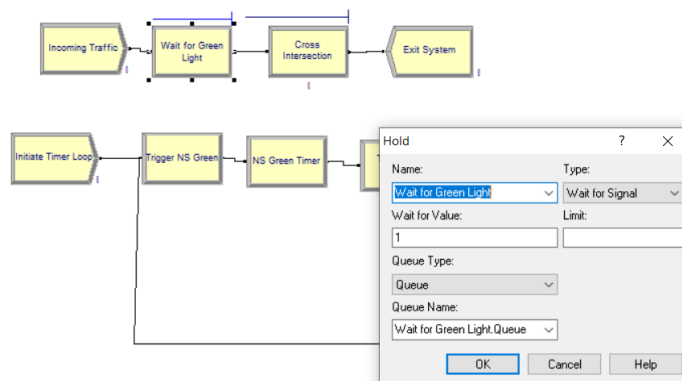
Figure 9.1: Simulation model of a traffic system

Configurations

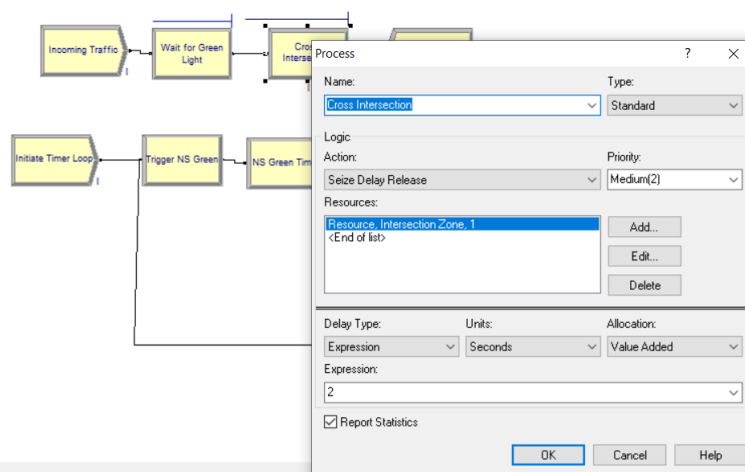
1. Incoming Traffic



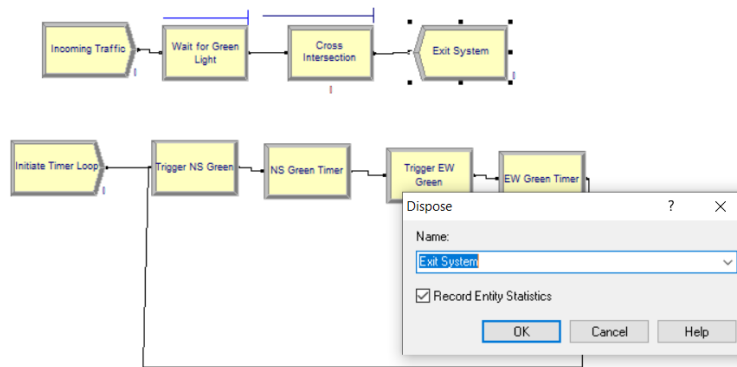
2. Wait for Green Light



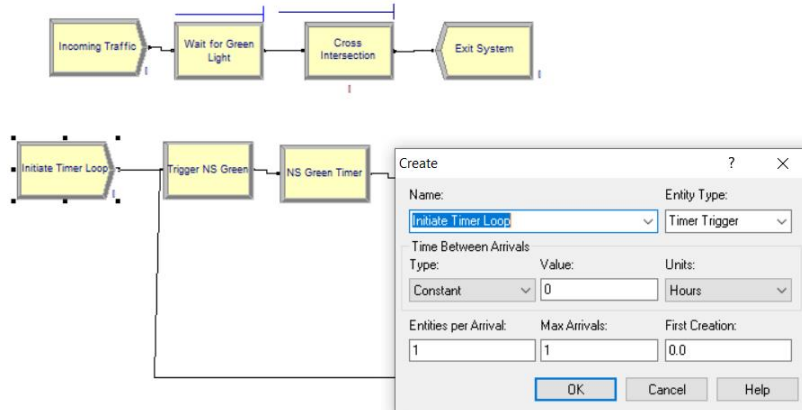
3. Cross Intersection



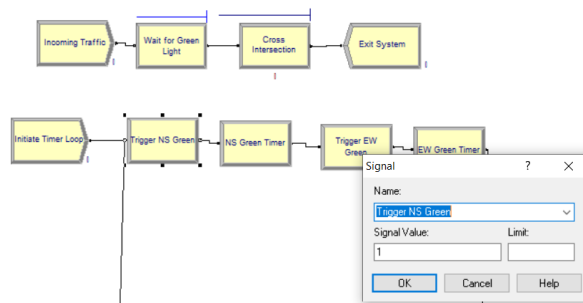
4. Exit System



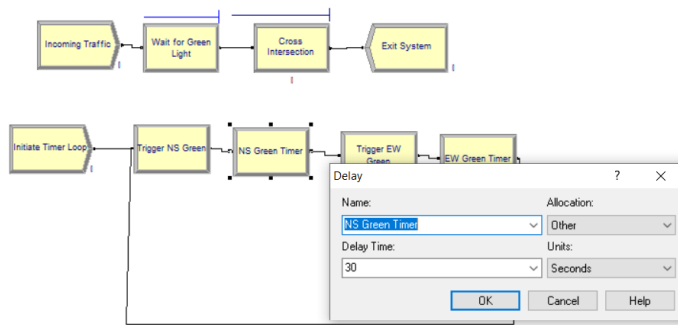
5. Initiate Timer Loop



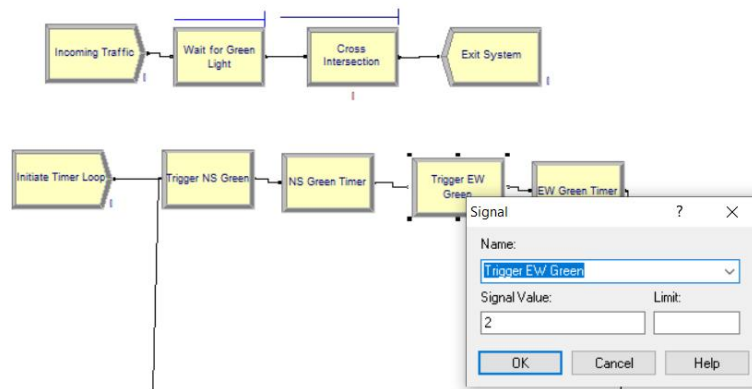
6. Trigger NS Green



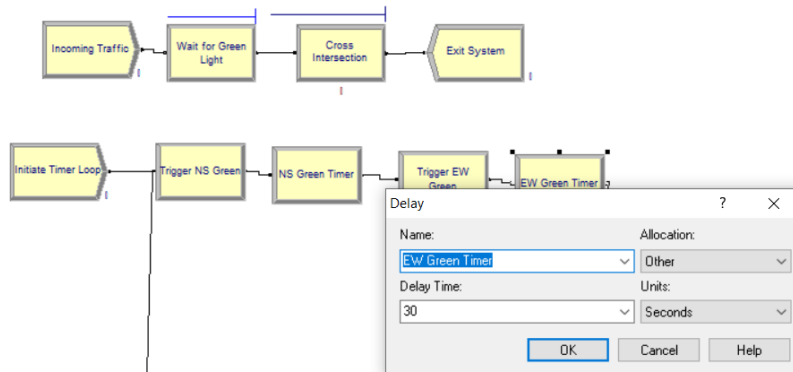
7. NS Green Timer



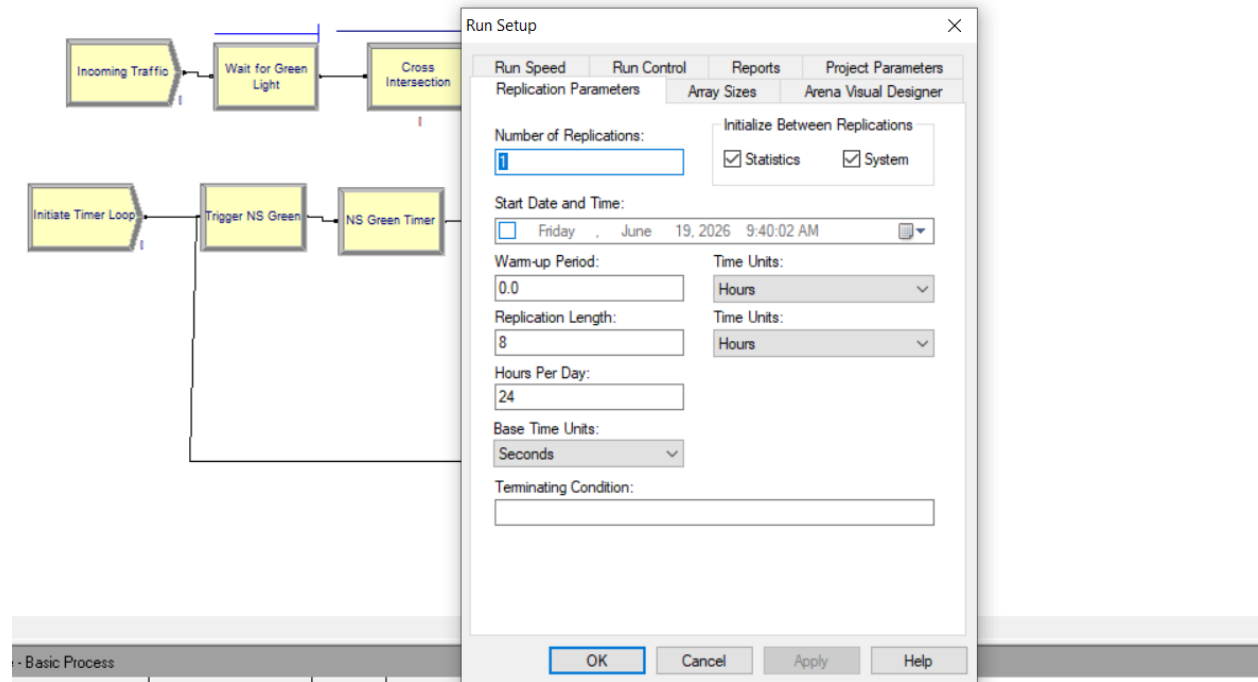
8. Trigger EW Green



9. EW Green Timer



10. Run Set-Up



Experiment 10: Simulation of work-force management by applying Agent-Based Modeling.

Objective

The objective of this experiment is to:

- Develop an agent-based simulation model that integrates deterministic work schedules with variable, fatigue-driven processing speeds to observe how individual operator dynamics influence overall call center throughput and system-wide efficiency.
- Collect and analyze critical Key Performance Indicators (KPIs)—specifically average customer wait times, maximum queue lengths, and operator utilization—to establish a quantitative baseline of how the system functions under varying levels of operational stress.
- Analyze how the combination of scheduled capacity loss and cumulative fatigue degradation influences the system's stability

Software

- Rockwell Arena Simulation/AnyLogic

Introduction

Beyond its capacity to simulate individual behavior, Agent-Based Modeling (ABM) is uniquely suited for addressing the non-linear challenges of workforce management. Traditional analytical models often assume steady-state productivity, yet in reality, workforce efficiency is rarely constant; it is subject to cycles of exertion, recovery, and interaction with organizational constraints.

ABM allows management researchers to move beyond these assumptions by capturing the feedback loops inherent in human labor—such as how a high-stress environment degrades individual performance, which in turn necessitates longer work hours, further increasing fatigue, and ultimately impacting total service quality. By modeling agents as entities that learn, adapt, and reach thresholds of burnout or fatigue, organizations can simulate "what-if" scenarios that reveal how seemingly minor policy adjustments, such as rescheduling break intervals or modifying shift lengths, can propagate through the entire system to prevent, or inadvertently trigger, operational collapse. This capability transforms workforce strategy from a reactive, descriptive exercise into a proactive, predictive one, offering insights into the tipping points of organizational stability that aggregate metrics simply cannot detect. Furthermore, the power of Agent-Based Modeling (ABM) lies in its ability to expose the stochastic nature of workforce performance. In traditional management models, labor is often treated as a homogeneous commodity with constant output. ABM, by contrast, acknowledges that workforce performance is highly sensitive to the internal states of individual operators—such as fatigue, psychological stress, and varying competence levels—which interact in non-linear ways.

Problem

A customer service center employs a fixed workforce of 5 operators during an 8-hour (480-minute) shift. Customers arrive at a predictable rate of one every 3.5 minutes, following an exponential distribution. The center uses a standard First-In-First-Out (FIFO) queue for all arriving customers. Each operator follows a fixed schedule: they work for 120 minutes and then must take a mandatory 15-minute break, effectively removing them from the queue-processing pool during that time. Operators start with a baseline processing speed of 3 to 5 minutes per request.

To track fatigue without complex loops, every completed call increments an operator's "Total Call Count" by one. The processing time for each new request is calculated as: $\text{Processing Time} = \text{Baseline Speed} \times (1 + \text{Total Call Count}/50)$. If an operator reaches a total of 50 completed calls, they are considered exhausted and are removed from the workforce for the remainder of the shift.

Construct the given system using Agent Based Modelling to collect data on average customer wait times, queue lengths, and operator utilization, and to analyze how the combination of scheduled breaks and the gradual degradation of processing speed influences the system's ability to handle the incoming call volume over the 480-minute period.

Model

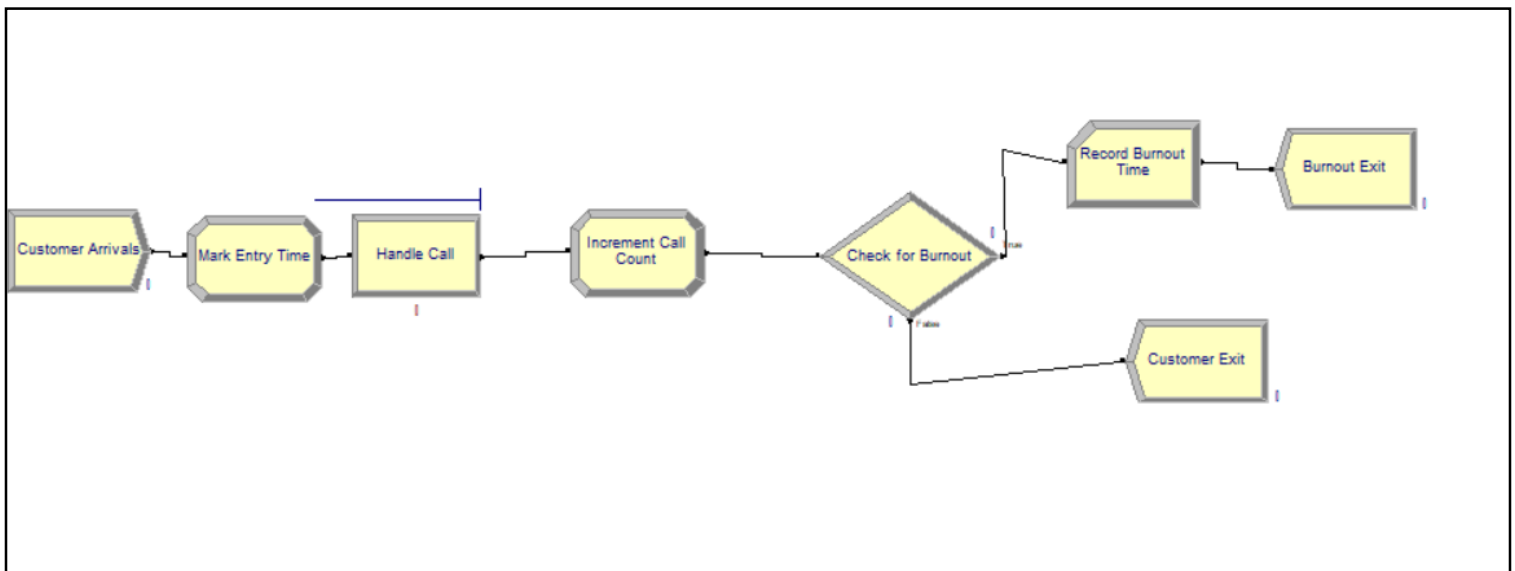
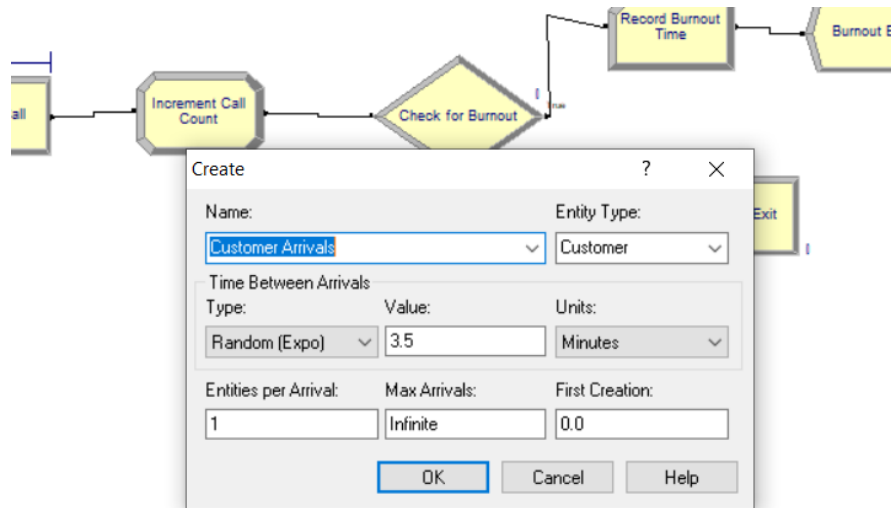


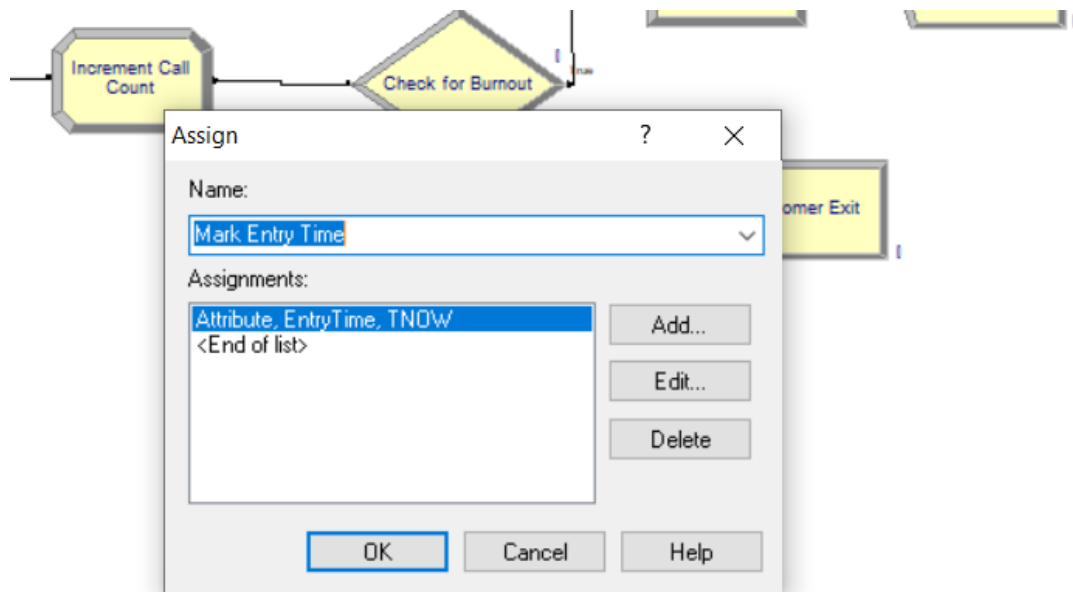
Figure 10.1: Agent Based Modelling Simulation Modelling for a customer service center

Configurations

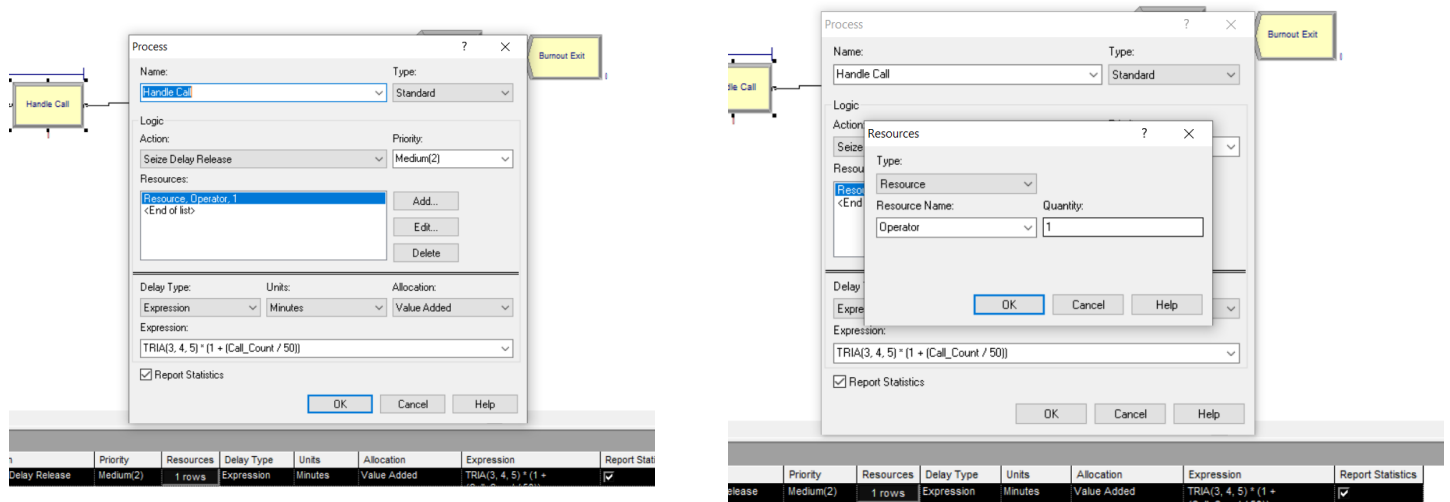
1. Customer Arrival



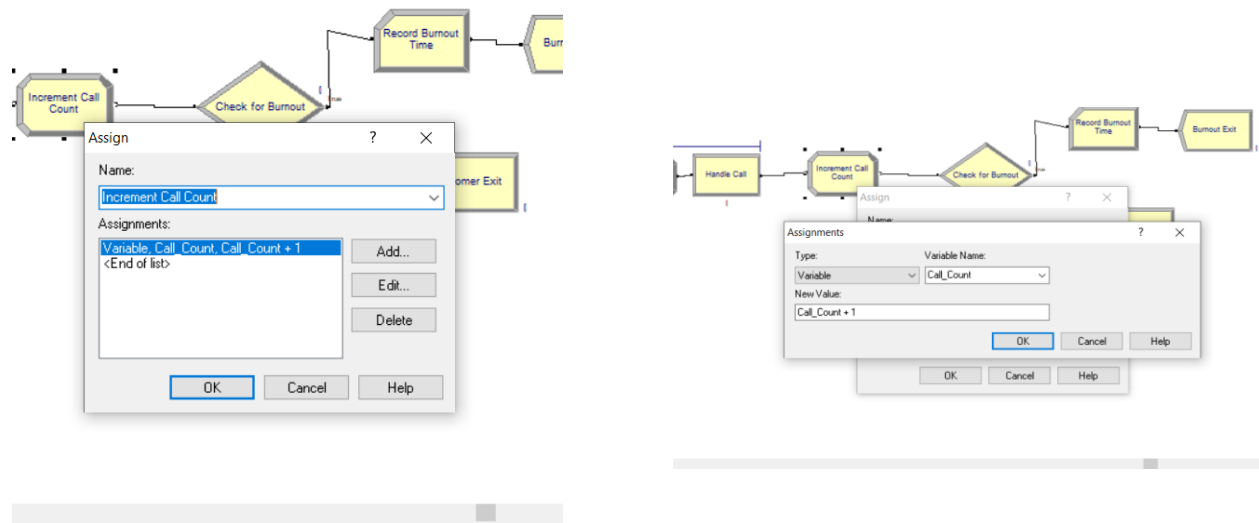
2. Mark Entry Time



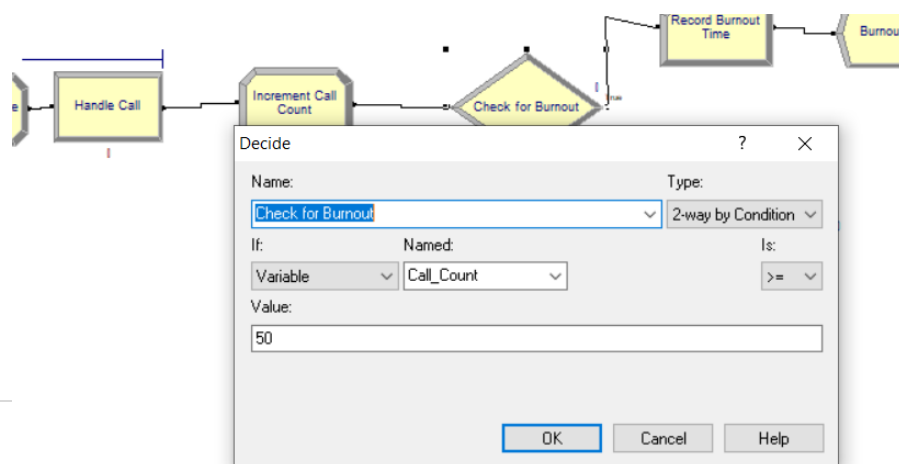
3. Handle Call



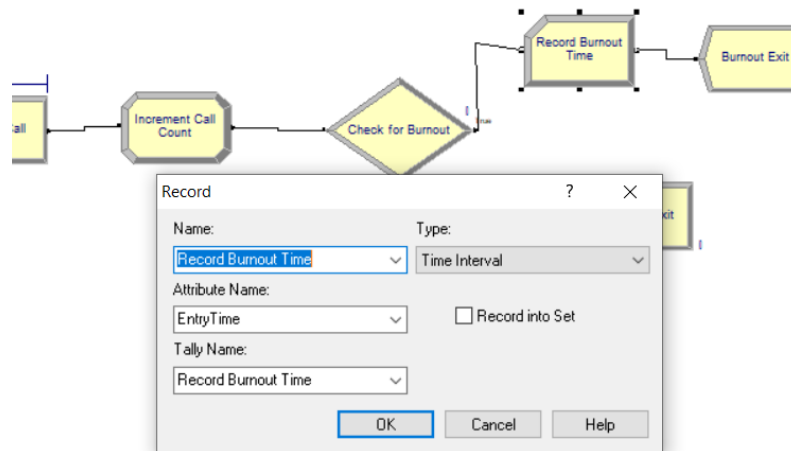
4. Increment Call Count



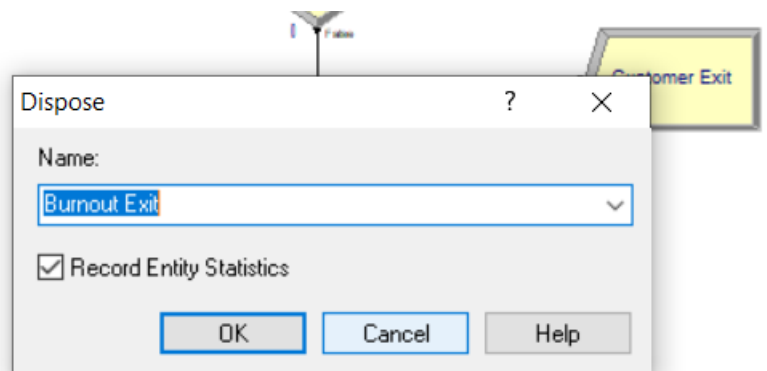
5. Check for Burnout



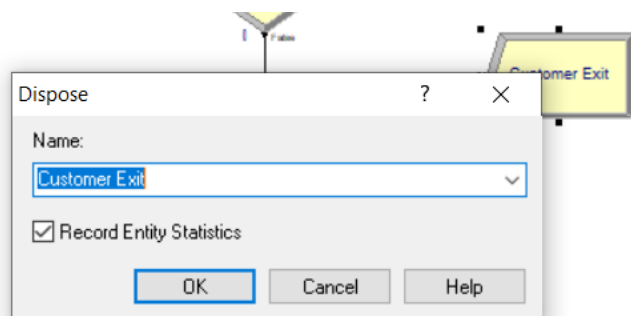
6. Record Burnout Time



7. Burnout Exit



8. Customer Exit



9. Schedule

The screenshot shows the 'Schedule - Basic Process' window. On the left is a sidebar with icons and labels: 'Schedule' (calendar icon), 'Process' (document icon), 'Scheduling' (document icon), 'Fact Data' (document icon), and 'Script' (document icon). The main area contains a table with the following data:

	Name	Type	Time Units	Scale Factor	Durations
1 ▶	OperatorSchedule	Capacity	Minutes	1.0	2 rows

Below the table, it says: "Double-click here to add a new row."

The 'Schedule' dialog box is open. It contains the following fields and controls:

- Name: FDM Info... (text field)
- OperatorSchedule (dropdown menu)
- Type: Capacity (dropdown menu)
- Time Units: Minutes (dropdown menu)
- Scale Factor: 1.0 (text field)
- Durations: A list box containing '5, 120', '0, 15', and '<End of list>'. The '5, 120' item is selected.
- Buttons: Add..., Edit... (highlighted), Delete
- Bottom buttons: OK, Cancel, Help

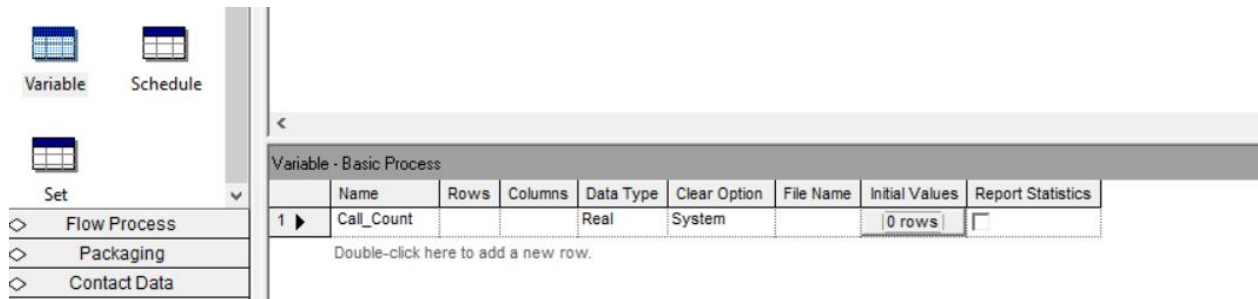
10. Resource

The screenshot shows the 'Resource - Basic Process' window. On the left is a sidebar with icons and labels: 'Resource' (calendar icon), 'Schedule' (calendar icon), 'Process' (document icon), 'Scheduling' (document icon), 'Fact Data' (document icon), and 'Script' (document icon). The main area contains a table with the following data:

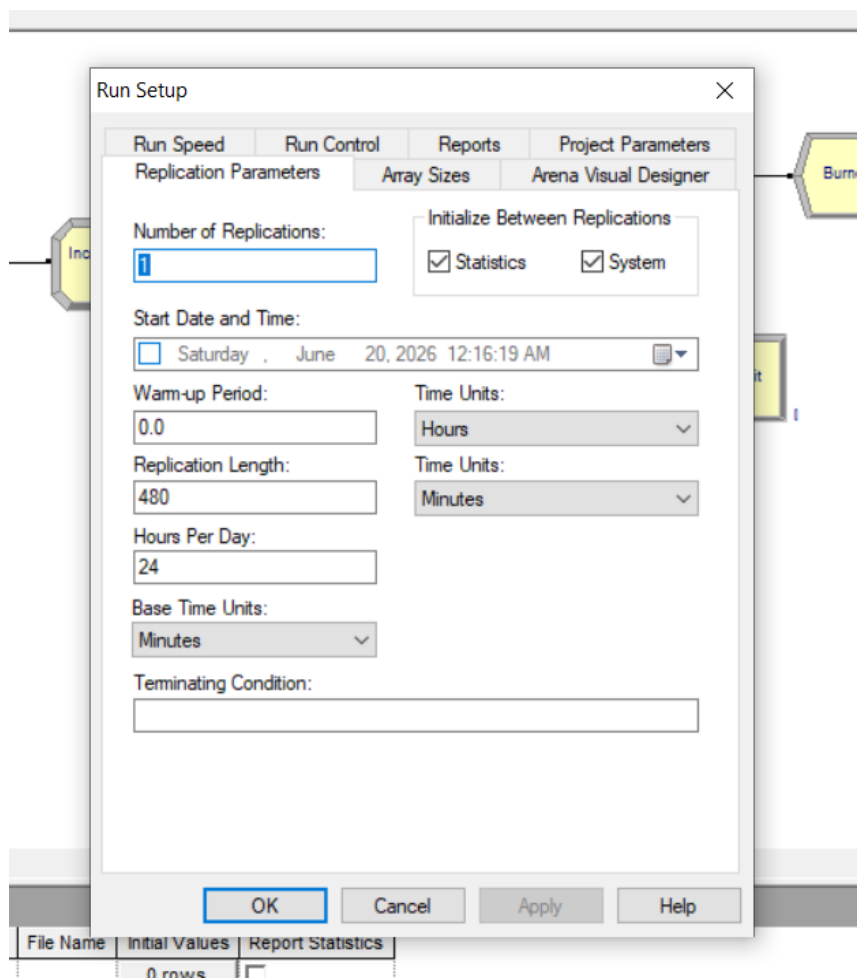
	Name	Type	Schedule Name	Schedule Rule	Busy / Hour	Idle / Hour	Per Use	StateSet Name	Failures	Report Statistics
1 ▶	Operator	Based on Schedule	OperatorSch	Wait	0.0	0.0	0.0		0 rows	✓

Below the table, it says: "Double-click here to add a new row."

11. Variable



Run Set-Up



Project

In the project part, students will have to design, implement, and analyze a simulation model to represent a real-world system under conditions of uncertainty. By applying techniques such as Monte Carlo methods, discrete event simulation, or agent-based modeling, they will translate complex operational challenges into computational frameworks to evaluate performance metrics and test the impact of various management policies. Ultimately, the goal is for students to transform raw data and logical constraints into a functional tool that provides actionable insights, allows for scenario testing, and supports data-driven decision-making.