

**School of Business Administration**

Operations Analytics - Course Project Report

Title : Amusement Park Simulation

Submitted By : Aswini Sivakumar

Samyuktha Kanakadandila

Ashreetha Udhayasankar

Submitted To : Dr. Osman T. Aydas

Date: 12/10/2023

TABLE OF CONTENTS

[**PROBLEM DESCRIPTION 3**](#_heading=h.7zthiej4cazf)

[**SIMULATION MODEL DETAILS 4**](#_heading=h.i1xhd5x4smp6)

[● Modeling of regular vs fast-pass customers and prioritization 4](#_heading=h.ba0dcnv4cnr0)

[● Modeling the Customer specifications 5](#_heading=h.up71ngvlqcn)

[● Modeling the customer walk times from the central node 6](#_heading=h.iln3051yo53v)

[● Modeling the ride area information 6](#_heading=h.f6lr9qn8hzue)

[● Modeling of wait lines, balking rules 8](#_heading=h.fay4hnvfoya3)

[● Modeling the customer capacity and buffer for other areas 8](#_heading=h.cq2bjo70myix)

[● Tracking of wait times for each customer type 9](#_heading=h.ehgr48ic24zg)

[● Modeling the satisfaction categories 10](#_heading=h.pxy5o5vltrs2)

[● Modeling the central node 11](#_heading=h.v1xcxvz058lm)

[● Modeling the conditions for bathroom visit, concession visit and arcade interest 12](#_heading=h.lcukikfrndeq)

[**EXPERIMENTAL DESIGN 14**](#_heading=h.fh703dy4nfog)

[● The average and maximum time in queue for both Fast Pass and Regular customers for the roller coaster and ferris wheel 14](#_heading=h.aj08m7pcdtd)

[● The average number of fast pass customers and regular customers in the amusement park 15](#_heading=h.8oe9kixquihm)

[● The average time a fast pass and regular customers waits in the queue of a ride 16](#_heading=h.3jkm0xsfdypc)

[● The average number of customers waiting and average time waiting at the bathroom 16](#_heading=h.uuyjico6acl3)

[● The percentage of satisfied regular and fast pass customers 17](#_heading=h.d7ccjoamwttl)

[**APPENDICES 18**](#_heading=h.35rl2mnihalq)

## PROBLEM DESCRIPTION

This project aims to develop a simulation model of an amusement park to evaluate waiting times and customer satisfaction as the proportion of Fast Pass priority customers is varied. The model is based on reference layout, area details and metrics provided for an amusement park facility.

The park is open for 12 hours each day and visitors spend time across various zones like Roller Coaster, Ferris Wheel, Arcades, Carnival Games, concession stand and bathroom area.

The key performance metrics to be evaluated are:

* Waiting times for Regular and FastPass customer types at different rides
* Customer satisfaction percentage - Computed based on threshold wait times at two key rides: Roller Coaster and Ferris Wheel. If a visitor waited below the threshold at BOTH these rides, they are considered satisfied.

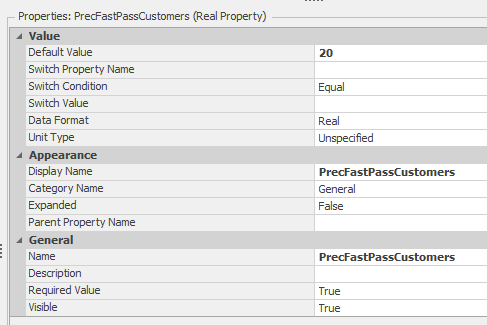
The base model will reflect current park demographics with 15% FastPass uptake. To study impact on above metrics, additional scenarios will be tested by increasing % of arriving customers purchasing Fast Pass from 10% to 20%

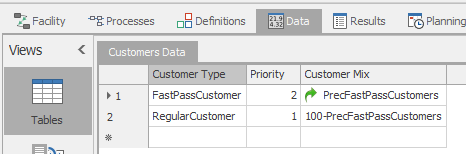
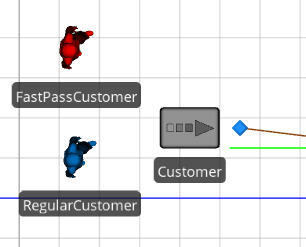
The model will be simulated for the park's operating duration of 12 hours. Each experimental configuration will undergo 100 replications with different random number streams to achieve statistical confidence.

## SIMULATION MODEL DETAILS

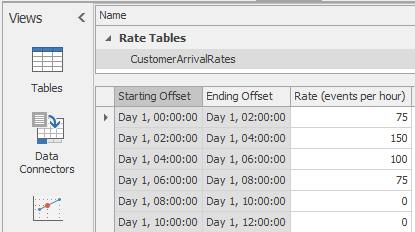
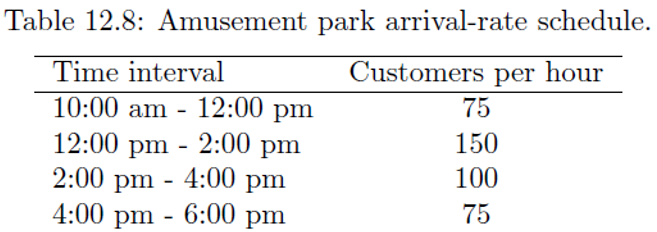
### Modeling of regular vs fast-pass customers and prioritization

After entering the park, customers have the opportunity to obtain the Fast-Pass, providing them with priority access to the Ferris Wheel and Roller Coaster. We have categorized our customers into two types: Regular Customers and FastPass Customers. The proportion of FastPass Customers in the system is controlled by a model property, **PrecFastPassCustomers**. The priority and customer mix for both types are defined in the **CustomersData** table, as mentioned below.



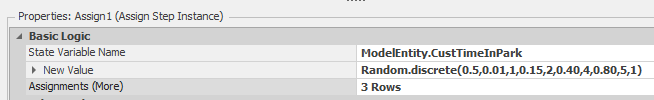


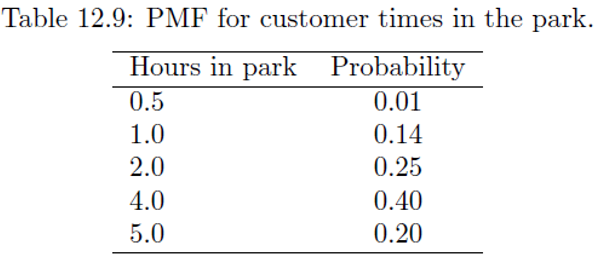
The problem description states that the customer arrival rate is a non-stationary Poisson process, as described in Table 12.8. We have modeled this process using the rate tables called **CustomerArrivalRates**. The simulation will start at the park's opening time at 10:00 AM. The first row of the rates table indicates that there will be 75 customers per hour for the first 2 hours, from 10:00 AM to 12:00 PM. Since customers will stop arriving at 6:00 PM, and the park will remain open until 10:00 PM, there will be no customers arriving during the last 4 hours. Hence, the last 2 rows of the rates table have values set to 0.



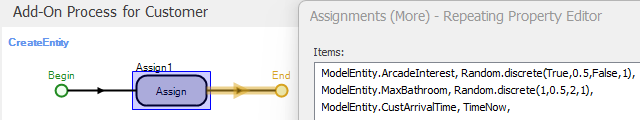
### Modeling the Customer specifications

To track the time spent by each customer in the park, we record their arrival time using a model entity state called '**CustArrivalTime**' in an add-on process named **'CreateEntity**.' The problem specifies the expected time spent by customers in the park as a probability mass function in Table 12.9. We utilize this function to determine the duration each customer will spend in the park by assigning the model entity state **'CustTimeInPark'** to the random discrete function, as illustrated in the screenshot below.



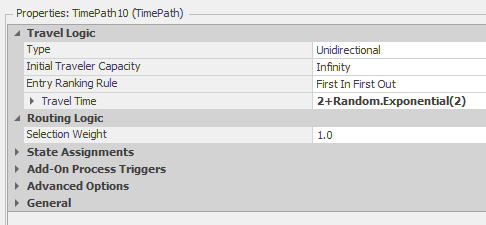
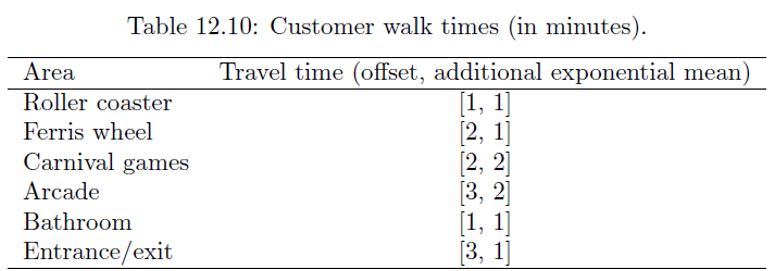


When creating each customer entity, we also specify their interest in the arcade and the maximum number of bathroom visits allowed, either once or twice during the visit. This is achieved by assigning respective values and expressions like mentioned below to the model entity states **'ArcadeInterest'** and **'MaxBathroom'**.



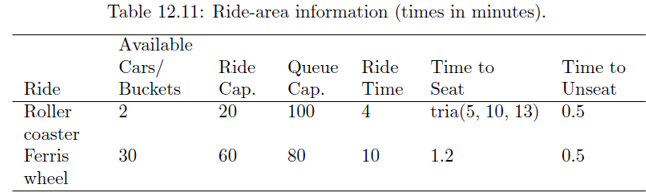
### Modeling the customer walk times from the central node

All customers will visit the central node to check the current waiting times for all rides and decide on their next ride. There is a walking time from the central node to each of the ride areas mentioned in the problem. These walk times are modeled using time paths from the central area to the respective rides. For instance, in the below screenshot, the **'Travel Time'** in Timepath10 represents the walk time for customers from the central node to the Carnival games area.

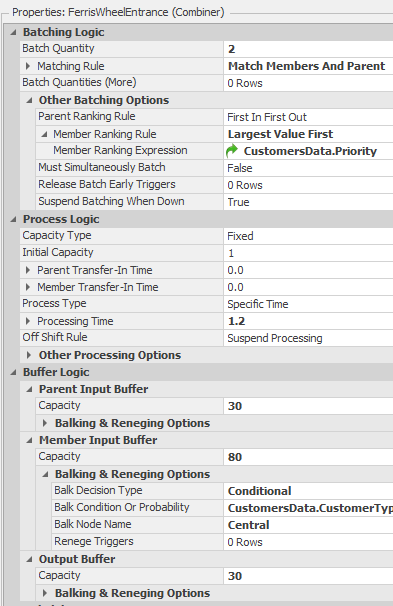
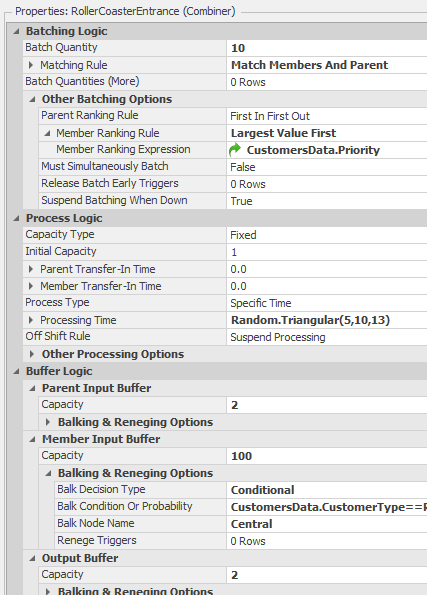


### Modeling the ride area information

The Ferris Wheel and Roller Coaster rides have specific capacity specifications outlined in Table 12.11. These rides are modeled using a combiner and a separator, with the parent being either a car (for the roller coaster) or a bucket (for the Ferris wheel), and the member being the customer. The ride capacity indicates the maximum number of customers each ride can accommodate. The queue capacity defines the maximum number of customers allowed to wait in line for a particular ride.



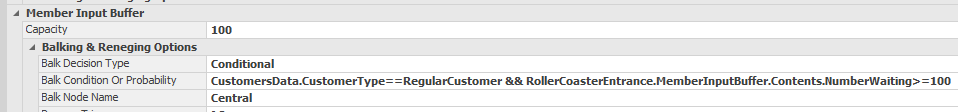
The available cars/buckets are specified in the capacity of the Parent Input Buffer of the respective combiners. The Batch Quantity denotes the number of customers each car/bucket can hold. For instance, the ride capacity of the roller coaster is 20, indicating that the number of cars represented by the Parent Input Buffer is 2, with a batch quantity of 10. The queue capacity is specified in the member input buffer capacity, and the time to seat for the ride is specified in the Processing Time of the combiner. Whereas, the time to unseat from the ride is specified in the Processing Time of the separator and the ride time for these two rides is mentioned in the Travel Time of the time path connecting the combiner and the separator.

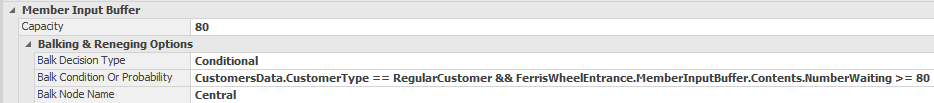


To model the priority that FastPassCustomers receive on these rides, we have specified the Member Ranking Rule as **'Largest Value First'** and included expression logic to retrieve the priority from the **CustomersData** table defined earlier.

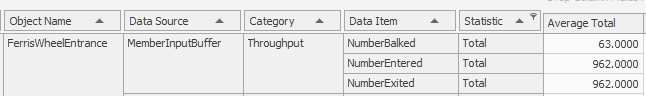
### Modeling of wait lines, balking rules

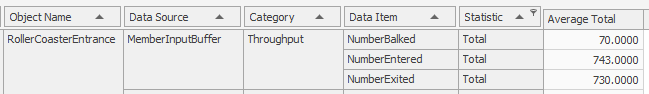
To model the avoidance of waiting in a long queue for RegularCustomers, we specified the balking condition for both the Roller Coaster and Ferris Wheel as mentioned below. This implies that if the number of customers waiting in line exceeds 100 in the case of Roller Coaster and 80 in case of Ferris Wheel, a regular customer arriving at the ride’s combiner will balk the line. In other words, they will not join the line and will instead return to the Central node to explore other viable options.





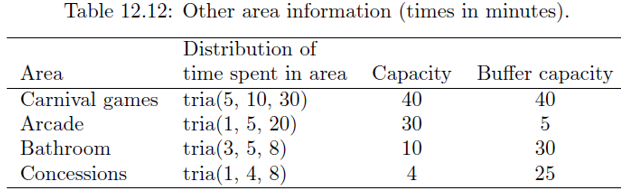
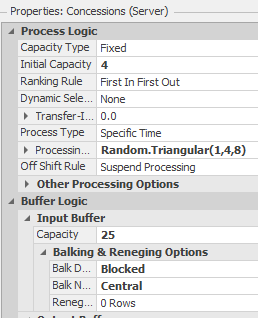
On average, as seen in the screenshots below, 63 customers balked the Ferris Wheel ride, while 70 customers balked the Roller Coaster ride.





### Modeling the customer capacity and buffer for other areas

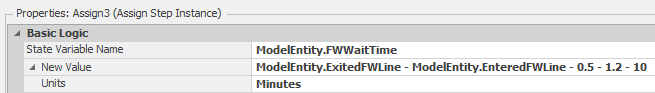
Table 12.12 provides information on the distribution of time spent, capacity, and buffer capacities of other areas in the amusement park. Each of the mentioned areas is modeled as a server, where the Initial capacity represents the area's capacity, and the Input Buffer capacity is the specified buffer capacity, as detailed in the table. For example, the screenshot below shows that the Concessions area has a capacity of 4, a buffer capacity of 25, and a processing time generated by a random triangular function (1,4,8). If there are 25 customers waiting in line, a new customer arriving at the concessions area will balk the line and return to the central node to explore other viable options.

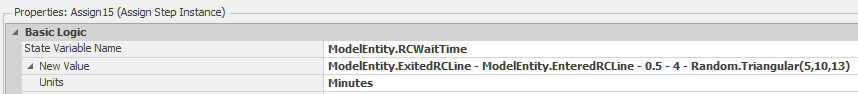


### Tracking of wait times for each customer type

The wait times for the Ferris Wheel and Roller Coaster rides are calculated using the model entity states FWWaitTime and RCWaitTime in an add-on process triggered at the member output node of the separators. The times when customers enter the ride are stored in the model entity state variable 'EnteredRCLine' for the roller coaster and 'EnteredFWLine' for the Ferris wheel. The times when customers exit the ride are stored in the model entity state variables 'ExitedRCLine' for the roller coaster and 'ExitedFWLine' for the Ferris wheel. The duration for which a customer waited for the ride is determined by the difference between the time the customer exited the line and the time the customer entered the line, with an additional delay accounting for the ride time, time to seat, and time to unseat.

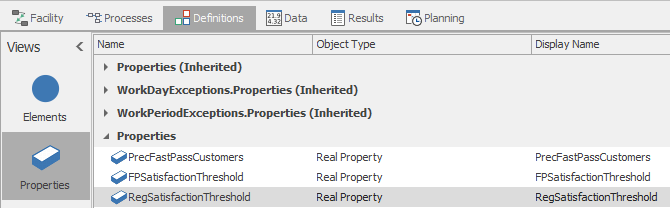
The below screenshots shows the wait time calculation for the customers at the Ferris wheel and Roller coaster.

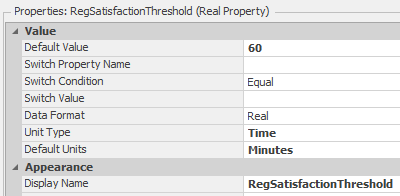
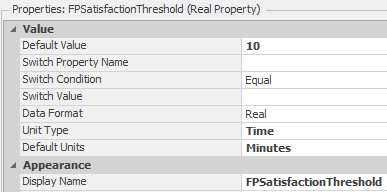




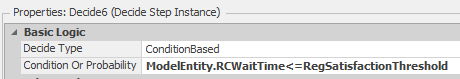
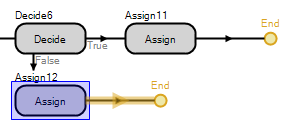
### Modeling the satisfaction categories

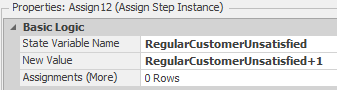
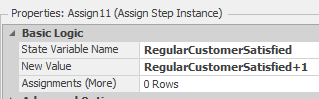
The problem description also specifies when a customer becomes discouraged due to excessive waiting times at the ride areas. To define the desired threshold until which customers are satisfied to wait, we created two model properties: **'FPSatisfactionThreshold'** for FastPassCustomers and **'RegSatisfactionThreshold'** for Regular customers. The screenshots below indicate that FastPassCustomers are willing to wait for 10 minutes, whereas regular customers are willing to wait for 60 minutes and become dissatisfied thereafter.





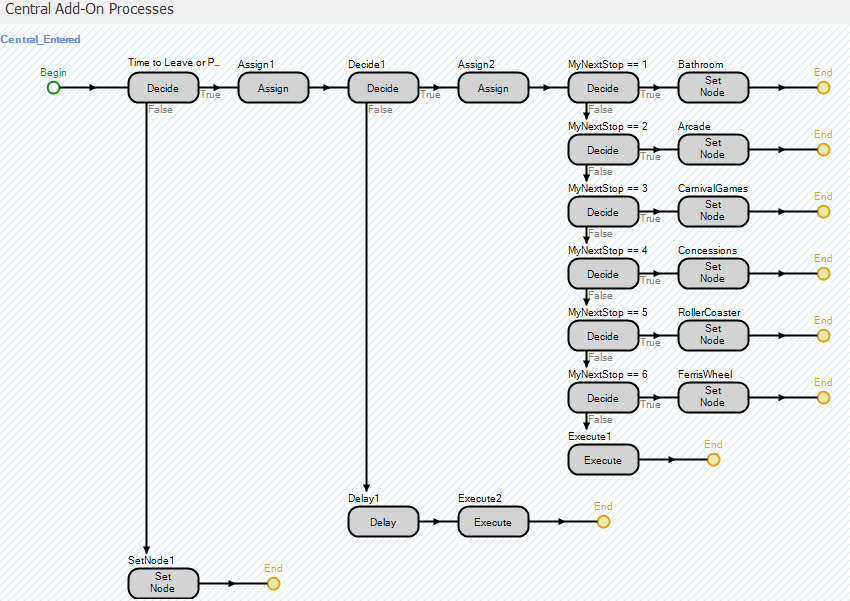
The number of satisfied and unsatisfied customers, whether regular or FastPass, is calculated using the model state variables 'RegularCustomerSatisfied,' 'RegularCustomerUnsatisfied,' 'FPCustomerSatisfied,' and 'FPCustomerUnsatisfied' in an add-on process triggered at the member output node of the separators. For example, in the case of the roller coaster, if the customer is a regular customer and the ride wait time for the customer is less than the 'RegSatisfactionThreshold,' then 'RegularCustomerSatisfied' is increased by a count of 1. And, if the condition is not true, then the 'RegularCustomerUnsatisfied’ is increased by a count of 1.

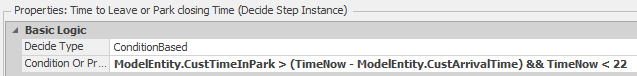




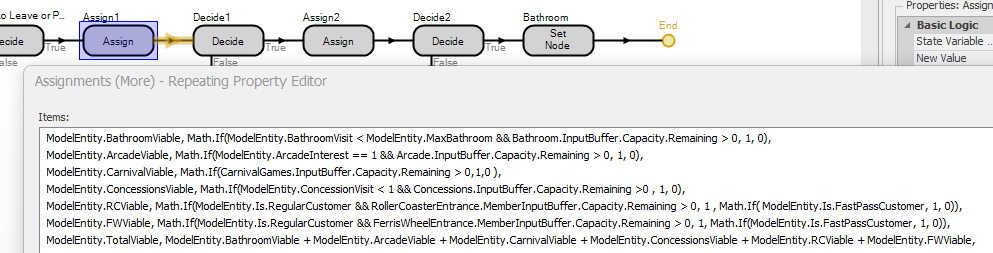
### Modeling the central node

The problem description states that customers visit the central node between each area visit to decide their next stop based on the waiting time for all the rides. Additionally, all areas are equally probable for each customer, provided their viable options at that point in time. For example, if a customer has already visited the concessions and is interested in the arcade, the options for their next stop in other areas like the Roller Coaster, Ferris Wheel, bathroom, arcade, and carnival games are equally probable. This decision making process is modeled using an add-on process ‘Central\_Entered’ and triggered when the customer enters the Central node.

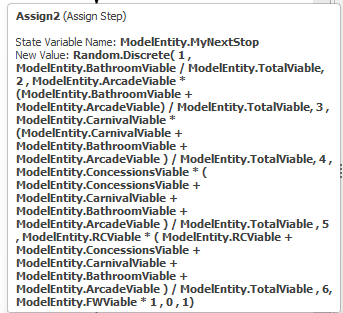




Once the customer enters the Central node, if the time spent so far is less than the assigned 'CustTimeInPark,' and the current time is before the park's closing time, our model identifies all viable options for the customer, as shown in the screenshot below. If either of the above conditions is not satisfied, the customer exits the park.

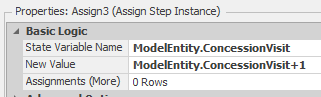


We then calculate the viability of each area for the customer and determine the total viable options available at the moment. To decide the next stop for the customer, we create a random discrete function, as shown below, and set the node to the next destination based on these probabilities.

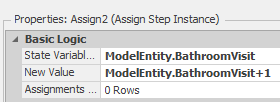
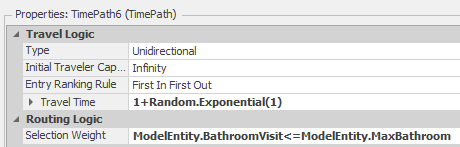


### Modeling the conditions for bathroom visit, concession visit and arcade interest

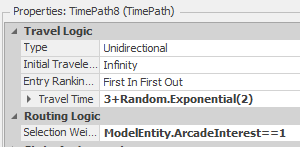
The problem description specifies that a customer can visit the concessions stand only once. To track the number of visits, we utilize a model entity state called 'ConcessionVisit' and implement an add-on process in the Concessions server. We incorporate this value in the central node process mentioned earlier to determine if the customer has already visited the concession stand, and consequently, we do not consider the Concessions stand as a viable option for their next stop.



The problem description also states that a customer can visit the bathroom either once or twice. We define this limit in the 'CreateEntity' process using a model entity state called 'maxBathroom' for each customer at the time of their creation. To enforce this limit, we incorporate the selection weight logic mentioned below to check if the customer has surpassed their bathroom visit limit for the day, allowing only those who have not exceeded it. Additionally, we use another model entity state, 'BathroomVisit,' to track the number of visits made by the customer to the bathroom and implement an add-on process in the Bathroom server.

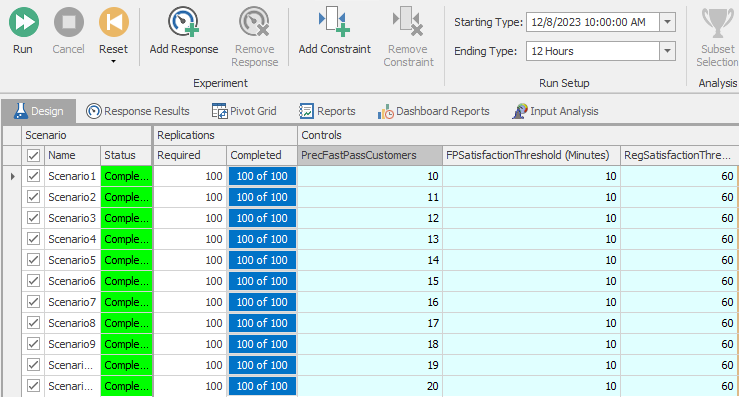


Similarly, the problem statement specified that only 50% of the customers are interested in the Arcade. We accomplished this by utilizing a model entity state called 'ArcadeInterest' in the 'CreateEntity' process, where we assign this value as 1 to only half of the created customers. In the time path from the central node to the Arcade, we implement the selection criteria mentioned below to allow only those customers who have 'ArcadeInterest' to access the arcade area.



## EXPERIMENTAL DESIGN

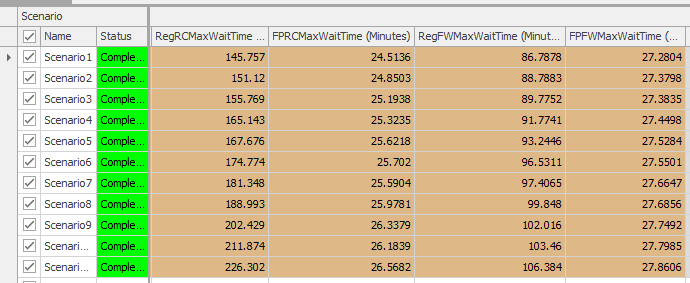
We ran the simulation for the amusement park spanning 12 hours, commencing at the park's opening time of 10:00 AM. Each scenario was replicated 100 times, with the 'PercFastPassCustomers' being controlled for each iteration.



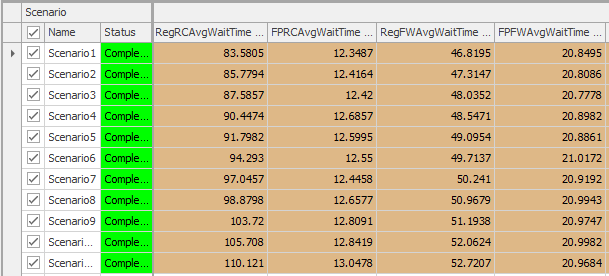
Here are a few of the statistics that we tracked in the experiments.

### The average and maximum time in queue for both Fast Pass and Regular customers for the roller coaster and ferris wheel

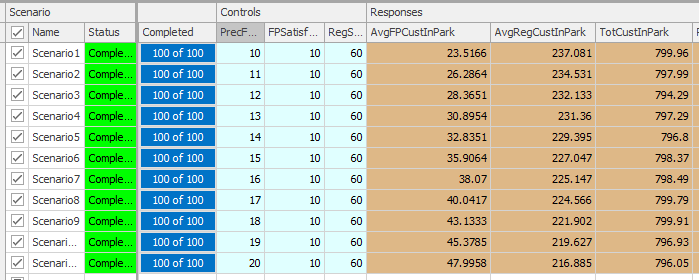
The response variables ‘RegRCMaxWaitTime’ and ‘FPRCMaxWaitTime’ represent the maximum waiting time for the regular and the FastPass customers for the Roller Coaster ride. Whereas, the response variables ‘RegFWMaxWaitTime’ and ‘FPFWMaxWaitTime’ represent the maximum waiting time for the regular and the FastPass customers for the Ferris Wheel ride.



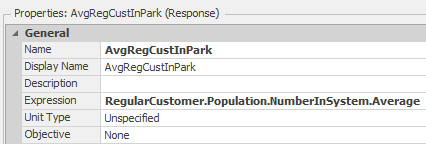
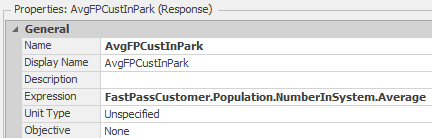
The response variables ‘RegRCAvgWaitTime’ and ‘FPRCAvgWaitTime’ represent the maximum waiting time for the regular and the FastPass customers for the Roller Coaster ride. Whereas, the response variables ‘RegFWAvgWaitTime’ and ‘FPFWAvgWaitTime’ represent the maximum waiting time for the regular and the FastPass customers for the Ferris Wheel ride.



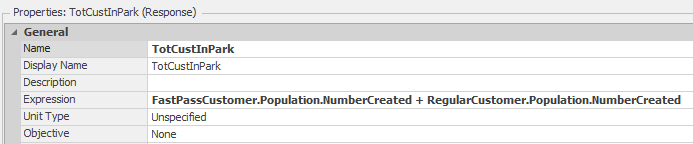
### The average number of fast pass customers and regular customers in the amusement park



The response variables 'AvgFPCustInPark' and 'AvgRegCustInPark' define the average number of FastPass and regular customers at any given point during the simulation period.

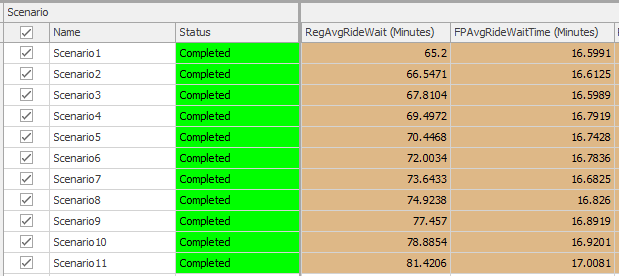


The total number of customers who visited the amusement park is defined by the response variable 'TotCustInPark.' It is evident that in all scenarios, the total number of customers is approximately 800, aligning with the theoretical value provided in Table 12.8.



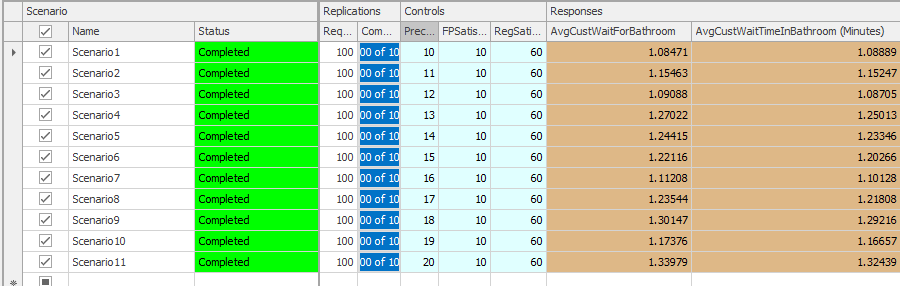
### The average time a fast pass and regular customers waits in the queue of a ride

The response variables ‘RegAvgRideWait’ and ‘FPAvgRideWait’ represent the average waiting time for the regular and the FastPass customers for the rides.



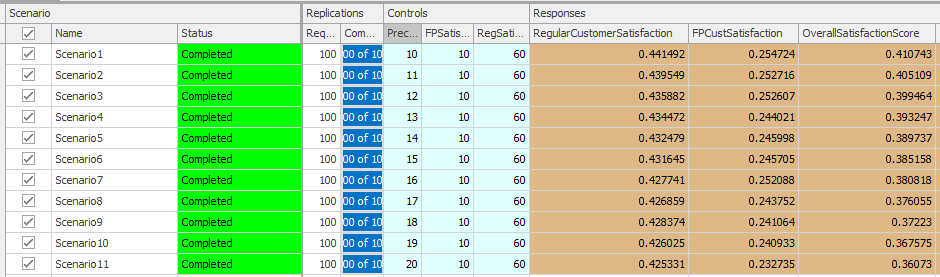
### The average number of customers waiting and average time waiting at the bathroom

The response variables ‘AvgCustWaitForBathroom’ and ‘AvgCustWaitTimeInBathroom’ represent the average number of customers waiting and the average waiting time in minutes at any given point in time during the simulation.

****

### The percentage of satisfied regular and fast pass customers

The response variables ‘RegularCustomerSatisfaction’ and ‘FPCustSatisfaction’ represent the percentage of regular and FastPass customers satisfied. The response variable ‘OverallSatisfactionScore’ represents the overall percentage of customers satisfied. The scenarios below show the values of customer satisfaction when the ‘PercFastPassCustomers’ ranges from 10 to 20.



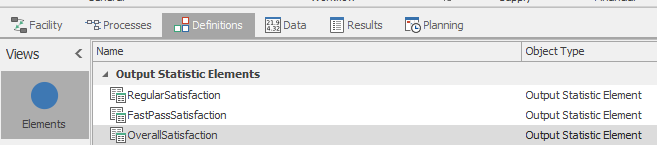
It is evident from the above table that customer satisfaction values decrease as the 'PercFastPassCustomers' is increased. This makes sense because increasing the percentage of FastPass customers sold will lead to longer waiting times for both regular and FastPass customers on the rides.

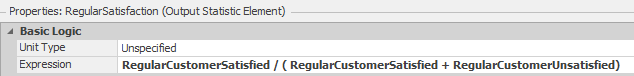
In conclusion, it is ideal for the park manager to limit the percentage of FastPass tickets to 10% to maintain a good satisfaction among all customers.

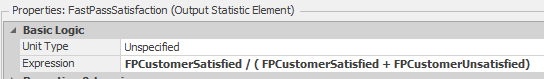
Overall, the model ran without any errors by considering all the specifications listed in the problem description and proved to be useful in computing all the statistics described above.

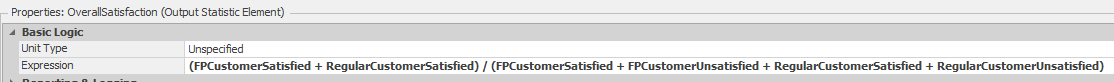
## APPENDICES

In order to obtain the satisfaction percentage mentioned above in the experiment, we created Output statistic elements ‘RegularSatisfaction’ , ‘FastPassSatisfaction’ and ‘OverallSatisfaction’ in the model.









The model entity states that were created in the process are as below.

