SM Theme Parks

CSI4124/SYS5110 – Foundations of Modeling and Simulation

Date

2019/01/16

Team Names:

Prem, David: CM

Salehi, Soroush: SM

Yang, Yilin: Team leader

Zhong, Fulin: CM

Zinaghaji, Hossein: SM

Table of Contents

Problem Description

Problem Statement

Bayou Adventure World is a theme park that has four distinct parks.

The main transportation option is the train, which travels in a loop between the four parks. We have a limited number of trains and cars per train, each of which has a maximum capacity of people they can carry.

The main objective is to minimize the cost of the train system, while also limiting the proportion of time that a people-mover leaves a station with passengers unable to board due to the train being full.

To evaluate this time, we define four types of performance categories:

Type 1 - Train leaves a stop with no people waiting to board,
Type 2 - Train leaves with 1 to 24 people still waiting,
Type 3 - Train leaves with 25 to 49 people still waiting, and
Type 4 - Train leaves with 50 or more people still waiting.

One of our main purposes is to minimize the percentage of time that people are waiting at the station. We believe that a properly designed system would have very few if any, Type 4 occurrences. We would also like to minimize the percentage of Type 2 and 3 occurrences.

Other objectives include:

1) Evaluating the impact of two loading/unloading options
2) Evaluating the impact of the new variable load time logic.

There are two possible options for boarding/deboarding, and the people-mover manufacturer has assured us that a new type of logic for boarding and deboarding trains (described in the SUI details) can be implemented. Still, we are unsure of its impact on the system.

Furthermore, another purpose is to keep our investment at a minimum. These costs include initial capital, operating costs, and maintenance costs for the projected life of each unit (trains/cars).

SUI Details

1. Park Layout

There will be several forms of transportation, including boats, a steam railroad, horse and wagon, and open-air buses. These transportation options are mostly theme-oriented and will only carry a small proportion of the people who will want to move between parks. The primary transportation option is to be a people-mover system. We describe here the people-move system.

1.1 General layout

The figure below shows that there are four stations in the Theme Park and a train system connecting them. The trains start at Frog Pond station and travel through Skunk Hollow station, Gator Island station, and Raccoon Corner station, and then back to Frog Pond station. The travel times are outlined in Figure 1, seen below:

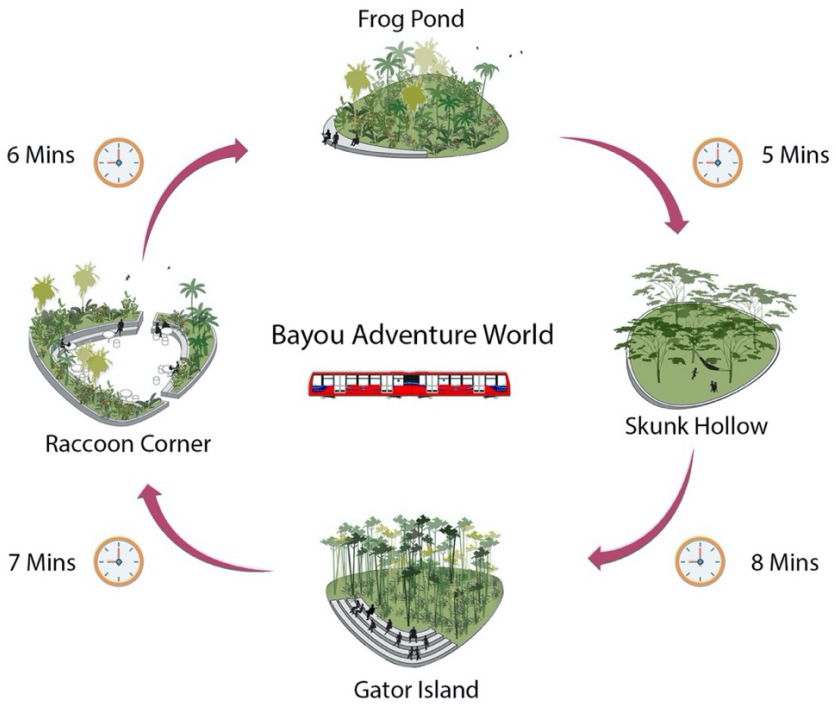

Figure 1: General layout and travel time of Bayou Adventure World

1.2 Trains

- This system can accommodate up to 8 trains. (***Figure 2***)
- The trains are computer-controlled and fully automatic, and each train consists of one or more cars, with each car having a capacity of a maximum of 25 people.
- The cost of each train includes initial capital cost, operating cost, and maintenance cost.
- The cost for the first car of each train is $800 per day. The cost for additional cars is $500 per day per car. (***Figure 2***)

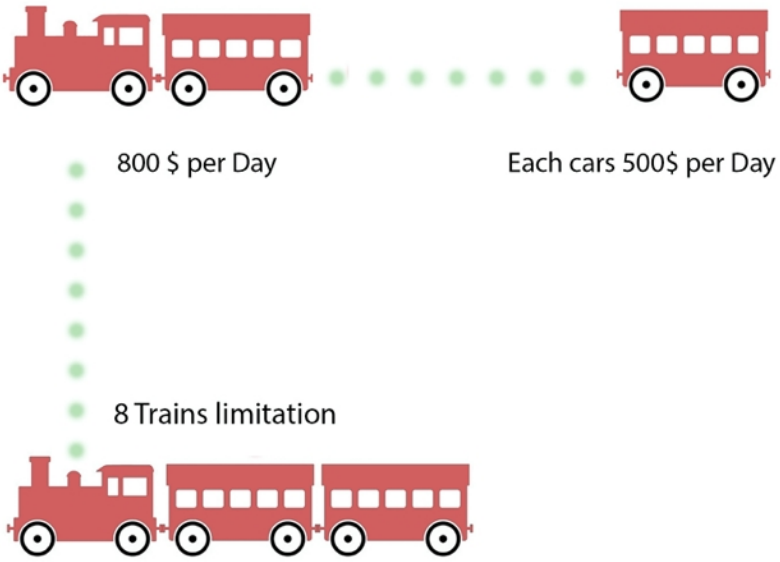

Figure 2: System limitation and cost of trains

1.3 Stations

Bayou Adventure World will contain one station in each of the following parks:

- Frog Pond
- Skunk Hollow
- Gator Island
- Raccoon Corner

Customers can enter Bayou Adventure World at any of the four parks and use the provided transportation to travel between them.

2. Transportation Service

2.1 Schedule Time

Bayou Adventure World is scheduled to be open between 10 AM and 10 PM daily. The people-mover manufacturer recommends that we start the trains a few minutes before we open the gates each day and run after the closing time for approximately one-half hour, or until all customers have departed the park.

2.2 Transportation loop:

The people-mover will be on a continuous loop: Frog Pond to Skunk Hollow to Gator Island in Raccoon Corner and back to Frog Pond, as shown in Figure 1

2.2.1 Travel times

Although there can be slight variations in travel time between parks, we can assume that travel times (in minutes), on average, are consistent. ***Travel times are listed below in Table 1.***

From	To	Travel Time (in min.)
Frog Pond	Skunk Hollow	5
Skunk Hollow	Gator Island	8
Gator Island	Raccoon Corner	7
Raccoon Corner	Frog Pond	6

Table 1: Travel times

2.3 Loading/Unloading Options

Two options exist for loading/unloading the trains, as shown below in Figure 3, the figure shows four options.

2.3.1 The First Option

The first option requires that customers board from one side and exit from the other. When a train stops at a station, the unload doors open for 30 seconds to allow passengers to exit. Then the load doors open for 45 seconds to allow new customers to board.

2.3.2 The second option

We plan to design the system to meet an above-average (but not peak) day of expected attendance. We will take advantage of well-below-average attendance days to conduct preventive maintenance on selected trains.

1) **First Basic Option, Two-sided doors:** Two-sided options are the most desirable option as it allows for a more orderly and controlled boarding and exiting process. However, this option costs an additional $20 per day per car. It is also possible that the second option might require additional cars. We would like to know the relative merits of these options.

2) **Second Basic Option, One side doors:** The second option uses a single side for both unloading and loading. The train stops at each station, the doors open, and customers are allowed to both board and exit at the same time. This option enables the doors to be open for 2 minutes.

3) **Alternate logic, variable boarding time:** The new logic would have the train stop for a reasonable time and depart if it were full, or almost full.
If the train is not full, it would wait to allow additional customers to board until the train became full, another train arrived at the station, or some reasonable period of time had elapsed.
The people-mover manufacturer has assured us that this type of logic can be implemented.

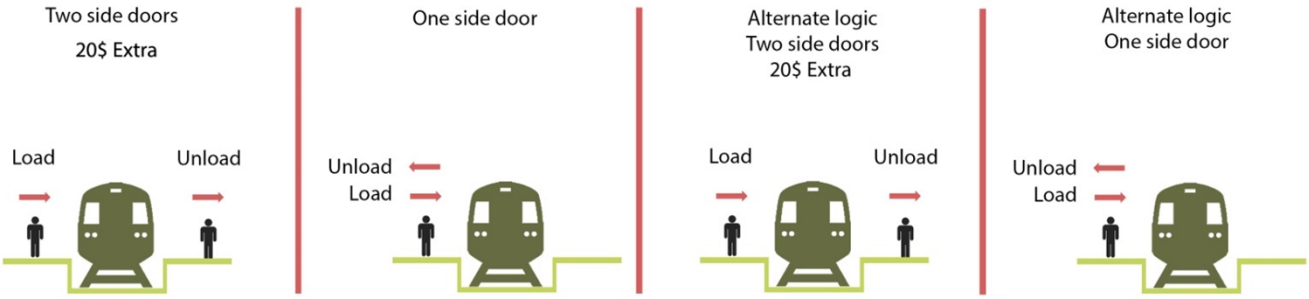

Figure 3: Four options

Two basic options exist for loading/unloading the trains. The basic options used a fixed time for loading/unloading. Alternative logic is available, which uses variable time for each of these two loading/unloading options. Figure three shows the four loading/unloading option.

Furthermore, we would always find that a few extra people will squeeze onboard instead of waiting for the next train. You also see situations where trains leave the station with empty seats, even though there is a line of people waiting to board. It is not clear what the actual capacity is under heavy-load conditions, but it should be close to 25.

2.3.3 The new logic

The new logic would have the train stop for a reasonable time and depart if it were full, or almost full.

If not, it would wait a short amount of time for additional customers. It would continue to load arriving customers until the train became full, another train arrived at the station, or some reasonable period of time had elapsed.

The people-mover manufacturer has assured us that this type of logic can be implemented.

Project Goal

The goal of the study is to determine the least cost train configuration (number of trains and number of cars per train, and which of the four boarding/unboarding options) that gives the desired customer satisfaction. The customer satisfaction level is measure in terms of the performance categories (type 1, 2, 3, and 4) presented in the Statement of the problem, more specifically the percentage of type 1, 2, 3, and 4 categories that occur during the day. We define the desired customer satisfaction level as follows:

- Percent of type 4 (more than 50 people at the station when the train leaves: 2%)
- Percent of type 3 (between 25 and 50 people at the station when the train leaves: 10%)
- Percent of type 2 (between 1 and 24 people at the station when the train leaves: 15%)

The cost per day of the configuration is computed as follows:

$$\textbf{cost} = \textbf{nT}*800 + \textbf{nC}*500 + \textbf{ts}*\textbf{nC}*20$$

where

nT is the number of trains
nC is the total number of cars
ts is 1 when using variable time boarding/unboarding option and 0 otherwise.

The parameters used to achieve this goal are:

numTrain: number of trains to solve the problem, which can have a value of one to eight trains.

numCar<T1,T2,…,T8>: Determine the number of cars per train with the minimum value of one and the maximum number of forty.

The maximum number of cars is estimated by the summing of arrivals in a rush-hour that is divided by two to be calculated for thirty minutes, which is the approximate travel time of a train in a complete cycle. Then it is divided by 25, which is the capacity of each car.

loadAndUnloadingOption: Indicates the way of boarding and deboarding to the train. This parameter can assume one of four values:

- **TWO_SIDE_DOORS:** The first option requires that customers board from one side and exit from the other.
- **ONE_SIDE_DOORS:** The second option uses a single side for both unloading and loading.

alternateLogic: When setting to TRUE, variable loading/unloading time is applied. When set to FALSE, fixed loading/unloading times are used. The parameter used with the loadAndUnlodingOption provides for the four possible loading/unloading options available.

The following solution strategy is used:

1. Initially, we are going to start with one train and fifteen cars and define the term train utilization as the percentage of time a train is full.
2. Run an experiment with the simulation model. Determine customer satisfaction levels.
3. If it did not meet our criteria, increase one to the number of cars.
4. We are going to repeat step two and three until it desired customer satisfaction level is met. Calculate and record the cost per day of the configuration.
5. Add one train and distribute fifteen cars into each train as evenly as possible.
6. Run an experiment with the simulation model, determine the customer satisfaction level as well as train utilization for each train.
7. If the desired customer satisfaction level is not reached, then increase the number of cars in the train that has the highest train utilization. If more than one train has the highest train utilization (e.g., 100%), then select the one with the least number of cars; if they have the same number of cars, then select one randomly.

8. Repeat steps six and seven until the desired customer satisfaction level is met. Calculate and record the cost per day of the transportation configuration.

9. Repeat step five to step eight until we obtain the daily cost of the transportation system by incrementing the number of trains by 1 up to 8. At the end of this step, 8 daily costs will have been determined during the experimentation, one for each possible number of trains in the configuration. The configuration with the lowest daily cost is selected as the configuration that potentially meets the project goal.

10. Repeat steps 1 to 9 for each of the loading/unloading options, which will give a least-cost train/car configuration for each of the four options. Select the configuration (train/car and loading/unloading option), which has the least cost as the solution to achieve the project goal.

Output

percentType1: percentage of departure occurrences that are of type 1 (train leaves with no people waiting to board).

percentType2: percentage of departure occurrences that are of type 2 (train leaves with 1 to 24 people waiting to board).

percentType3: percentage of departure occurrences that are of type 3 (train leaves with 25 to 49 people waiting to board).

percentType4: percentage of departure occurrences that are of type 4 (train leaves with 50 or more people waiting to board).

trainUtilization: Output provides a vector of numTrain values (<T1, T2, ...>, which gives the train utilization for each train in the system. Train utilization is defined as the percentage of time a train is full.

Study: Bounded Horizon Study.

Observation Interval:

- Time units are hours.
- 10 AM to the time when all customers have departed after 10 PM (i.e., all trains are empty).

ABCmod Conceptual Model

High-Level Conceptual Model

Simplifications

- We are going to assume that the capacity of each car is 25. We are not going to consider if there are few empty seats available or some squeezed people on board.
- The 10 seconds delay due to the blocking of the door by a customer in the second option is not going to be considered in our simulation because We are not explicitly modeling the passengers in our project. Furthermore, this amount of time is negligible.
- Cars are not going to be modeled as explicit entities but as the capacity of the trains.

Assumptions

- The reasonable waiting time in the new logic is going to be considered the same as the loading and unloading options. The related time for the ONE_SIDE_DOOR is one minute and fifteen seconds, and for the Two_SIDE_DOOR is two minutes.

Structural View

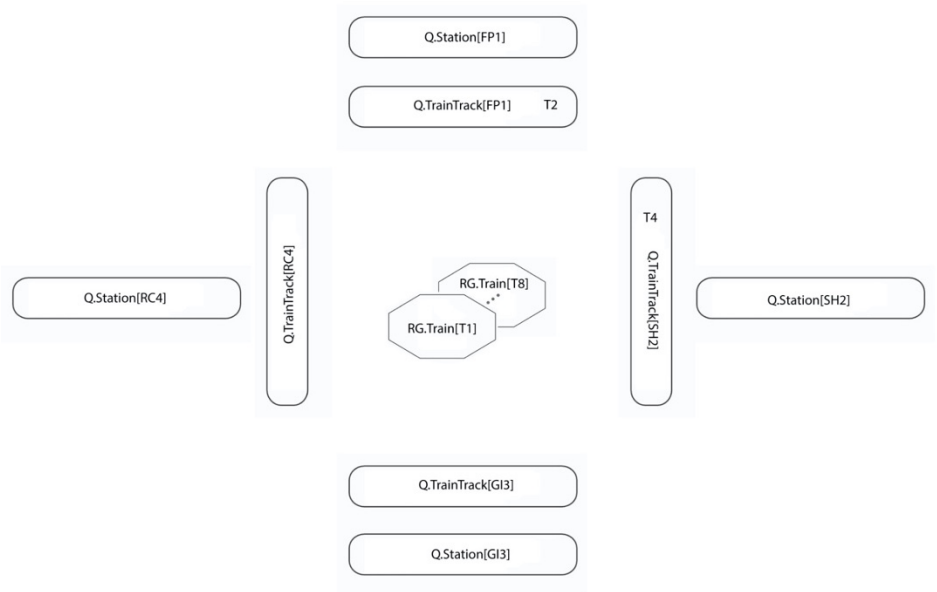

Figure 4: General Structural View

Entity categories:

- **RG.Trains**: An entity structure with **role=Resource Group** and **scope = Many[numTrains]** that represent the train in which passengers are being transported to four different stations. We use T1, T2 up to T8 to represent the member identifiers of the category which has scope = Many[N] where the maximum N = 8. Thus the entities representing the trains are RG.Trains[T1], RG.Trains[T2] up to RG.Trains[T8].
  We represent the number of passengers on the train using the attribute **numPassengers** for the train entities which is a four-value vector **<FP, SH, GI, RC>** where each of the values represents the number of passengers disembarking the train at Frog Pond, Skunk Hollow, Gator Island, and Racoon Center respectively. The cars of the train are represented using the attribute capacity, which is given the value of 25***numCars[ID]** was **ID** is the index to the corresponding train.
- **Q.Stations:** A set of four queues that represent the four stations that could contain the number of passengers.We use FP, SH, GI, and RC to identify each station, which has role = Queue, scope = Many[4].
  The number of passengers waiting to board a train is given by the standard attribute "n," and that there are no explicit entities added to the queue, that is, the standard attribute "list" is not used. eg: Q.Stations[FP1].n. The queue Q.Stations[FP1] represents the station Frog Pond, which could effectively have a large capacity of customers. The stations, Q.Sations[SH2], Q.Stations [GI3], and Q.Stations [RC4] represent the four stations(Frog Pond, Skunk Hollow, Gator Island, Raccoon Conner) respectively.
- **Q.TrainTrack：** A set of four queues that represent the track of each station. The train, which is located in the head of the queue, would be the first to load, unload, and depart. We could represent this by using the identifier **Q.TrainTrack[T1]**

Behavioral View

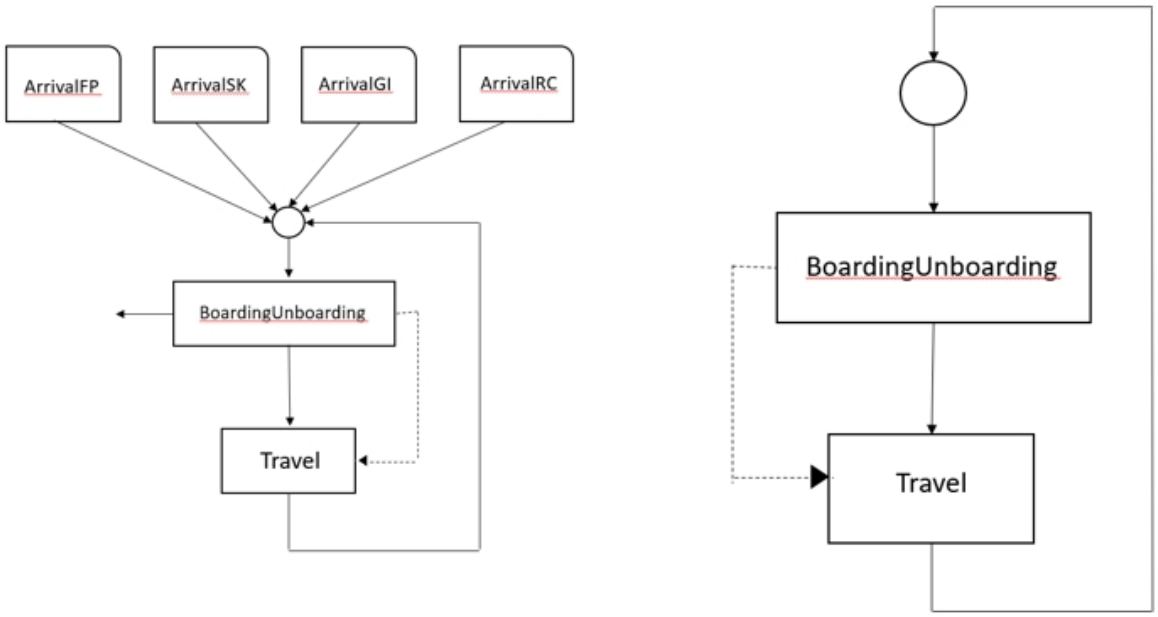

Figure 5: Behavioural diagram (a) Passenger lifecycle (b) Train lifecycle

Scheduled Actions Constructs:

Arrivals: the arrival of customers to one of the train stations

Extended Conditional Activity Constructs:

BoardingUnboarding: Customer boards or unboards the train.

Sequel activity:

Travel: This activity moves the train from one station to the next.

Input

Exogenous Input (Entity Stream)			
Variable Name	Description	Domain Sequence	Range Sequence
uPassangerArrFP uPassangerArrGI uPassangerArrSH uPassangerArrRC	This input entity stream represents the arrivingcustomers in each station	RVP.upassangerArr(RC) RVP.upassangerArr(SH) RVP.upassangerArr(GI) RVP.upassangerArr(FP)	1 passanger arrives at each arrival time for each station.
Endogenous Input (Independent)			
Variable Name	Description	Value(s)	
uTravelTime	This input represents the travel time of the trains between two stations.	DVP.uTravelTime (StationID)	
uLoadingUnloadingTime	This input represents the boarding time when the trains arrive at the stations.	DVP.uBoardingTime()	

Detailed Conceptual Model

Structural Components

Constants		
Name	**Description**	**Value**
FP1, RC2, SH3, GI4	An identifier for the members of the entity categories Q.Station and Q.TrainTrack, which have scope = Many[4].	Frog Pond station, Skunk Hollow station, Gator Island station, Raccoon Conner station
T1, T2, T3, T4, T5, T6, T7, T8	The identifier represents the members of the entity categories, which has scope = Many[8].	Train number 1, Train number 2, up to Train number 8
ID	The index to the corresponding train	0 up to 7
Parameters		
Name	**Description**	**Value**
numCars	The number of cars for each train	The minimum value of one and the maximum number of forty. 1 up to 40
numTrains	The number of trains that traveling among the four stations	1 up to 8
loadAndUnloadingOption	Indicates the way of boarding and unboarding to the train.	This parameter can assume one of four values, ○ **TWO_SIDE_DOORS:** The first option requires that customers board from one side and exit from the other. ○ **ONE_SIDE_DOORS:** The second option uses a single side for both unloading and loading.
alternateLogic	When set to **TRUE**, variable loading/unloading time is applied. When set to **FALSE**, fixed loading/unloading times are used	**TRUE** **FALSE**

Consumer Transient: passengers	
represent the number of passengers on the train using the attribute numPassengers for the train entities	
Attributes	**Description**
numPassengers	The assigned value indicates the number of passengers

Resource group Many[numTrains]: Trains
The number of trains that travel in the model. Their respective identifiers are the constants T1, T2 up to T8.

Queue Many[4]: Stations
A set of four queues that represent the four stations that could contain the number of passengers.

Attributes	Description
N	The number of passengers waiting to board on a train
Queue Many[4]: TrainTrack	
A set of four queues that represent the track of each station.	
Attributes	**Description**
N	The number of trains waiting to depart

Behavioral components

Output

Outputs	
Simple Scalar Output Variable (SSOV)	
Q.Stations[id].n	**SSOV Variable**
0	SSOV.numType1
1 to 24	SSOV.numType2
25 to 50	SSOV.numType3
More than 50	SSOV.numType4
percentType1	SSOV.numType1/($\sum_{i=1}^{i=4}$SSOV.numType i)
percentType2	SSOV.numType2/($\sum_{i=1}^{i=4}$SSOV.numType i)
percentType3	SSOV.numType3/($\sum_{i=1}^{i=4}$SSOV.numType i)
percentType4	SSOV.numType4/($\sum_{i=1}^{i=4}$SSOV.numType i)

Trajectory Sequences	
Name	**Description**
TRJ[TrainFull[id]]	Trajectory sequence reflects the time that RG.Train[id].capacity = RG.Train[id].numPassengers, which has the value of 1, otherwise it is 0.

Derived Scalar Output Variables (DSOV's)			
Name	**Description**	**Data Sequence Name**	**Operator**
trainUtilization[id]	Percentage of time that train[id] is full.	TRJ[TrainFull[id]]	Average

User-Defined Procedures

User Defined Procedures	
Name	**Description**
UDP.StationBoardingUnboarding()	Find id of the station where boarding/unboarding can start under following conditions: 1) Train is at the head of the station train track. (Q.trainTrack[id].n ≠ 0) 2) Not boarding (RG. Train[Q.trainTrack[id].list[0]].boarding = False)

<table>
<tr><td>UDP.BoardTrain(stid, tid)</td><td>The number of persons that board train is RG.Train(tid).Capacity – (Sum of all passengers in numPassengers attribute.)
The number that can board is as distributed as follows:

<table><tr><th rowspan="2">FROM</th><th colspan="4">TO</th></tr><tr><th>Frog Pond</th><th>Skunk Hollow</th><th>Gator Island</th><th>Raccoon Corner</th></tr><tr><td>Frog Pond</td><td>—</td><td>40</td><td>35</td><td>25</td></tr><tr><td>Skunk Hollow</td><td>37</td><td>—</td><td>39</td><td>24</td></tr><tr><td>Gator Island</td><td>42</td><td>29</td><td>—</td><td>29</td></tr><tr><td>Raccoon Corner</td><td>41</td><td>28</td><td>31</td><td>—</td></tr></table></td></tr>
<tr><td>UDP.GetDestination(stid)</td><td>Gives the id of the destination station:

<table><tr><th>Origin (stid)</th><th>Destination (stid)</th></tr><tr><td>FP</td><td>SH</td></tr><tr><td>SH</td><td>GI</td></tr><tr><td>GI</td><td>RC</td></tr><tr><td>RC</td><td>FP</td></tr></table></td></tr>
</table>

Input Constructs

Action: PassengerArrivals (RC)	
The arrival of a Passenger.	
TimeSequence	RVP.upassangerArrRC()
Event	iC.Passenger ← SP.Derive(Passenger) SP.InsertQue(Q.station[RC],iC.Passenger)
Action: PassengerArrivals (SH)	
Arrival of a Passenger.	
TimeSequence	RVP.upassangerArrSH()
Event	iC.Passenger ← SP.Derive(Passenger) SP.InsertQue(Q.station[SH],iC.Passenger)
Action: PassengerArrivals (GI)	
Arrival of a Passenger.	
TimeSequence	RVP.upassangerArrGI()
Event	iC.Passenger ← SP.Derive(Passenger) SP.InsertQue(Q.station[GI],iC.Passenger)
Action: PassengerArrivals (FP)	
Arrival of a Passenger.	
TimeSequence	RVP.upassangerArrFP()

Event	iC.Passenger ← SP.Derive(Passenger) SP.InsertQue(Q.station[FP],iC.Passenger)	
Random Variate Procedures		
Name	**Description**	**Data Model**
uPassengerArrFP() uPassengerArrGI() uPassengerArrSH() uPassengerArrRC()	Returns the next arrival time for a passenger, based on the assumption that t an arrival has occurred at current time t.	t + Exponential(MEAN) where MEAN is the everage of time period in each station.
Deterministic Value Procedure		
Name	**Description**	**Data Model**
uTravelTime(stationID)	Returns the traveling time for a station.	There is a fixed traveling time for each station. FP->SH = 5mins SH->GI = 8mins GI->RC = 7mins RC->FP = 6mins
uBoardingUnboardingTime()	Returns the boarding and unboarding time of trains arrive in stations.	The related time for the option ONE_SIDE_DOOR is one minute and fifteen seconds, and for the option Two_SIDE_DOOR is two minutes.

Behavioral Constructs

Activity: BoardingUnboarding	
Time at which passengers may board or unboard the train.	
Precondition	UDP.StationBoardingUnboarding() ≠ NONE
Event SCS	stid ← UDP.StatioBoardingUnboarding() tid ← Q.Station[stid].list[0] RG.Train[tid].boarding ← TRUE
Duration	DVP.uBoardingUnboardingTime()
Event SCS	UDP.BoardTrain(stid, tid) SPstartSequel(Travel, tid, stid)
Activity: Travel	
The train travels in between stations.	
Precondition	RG.Train[tid].boarding = TRUE AND (Q.TrainTrack[stid].N > 1 OR RG.Train[tid].numPassengers = RG.Train[tid].capacity)
Event SCS	tid ← SP.RemoveQue(Q.TrainTrack(stid)) dstid ← UDP.GetDestination(stid) RG.Train[tid].boarding ← FALSE
Duration	DVP.uTravelTime(stid)

Event SCS	SP.InsertQue(Q.TrainTrack[dstid])

Annex A-Data Modelling

(1) Customer arrivals:
Our projected customer arrivals (arrivals per hour) to each station are given below. Although there will probably be more variations, we can only provide hourly rates at this time.
The following Table 1 shows the number of customers who arrive at a station (e.g., Frog Pond station) and want transportation to another station.

Time Period	STATION			
	Frog Pond	**Skunk Hollow**	**Gator Island**	**Raccoon Corner**
10 AM - 11 PM	450	400	325	385
11 AM - 12 PM	300	350	340	320
12 PM - 1 PM	275	250	260	280
1 PM - 2 PM	285	275	210	265
2 PM - 3 PM	310	290	240	290
3 PM - 4 PM	320	305	280	315
4 PM - 5 PM	280	300	290	300
5 PM - 6 PM	260	280	275	320
6 PM - 7 PM	290	310	295	280
7 PM - 8 PM	315	320	330	310
8 PM - 9 PM	385	360	395	360
9 PM - 10 PM	415	405	430	395

Table 1: The number of customer arrivals

(2) Customers boarding percentage:
The number of customers who can board a train at any station depends on the capacity of each train and the number of current occupants. Thus, we realize that there must be a mechanism for determining the number of customers exiting at each station. Table 2 below provides these data in terms of the expected percent of customers who will exit each station based on where they boarded. For example, 39% of the customers who board the train at Skunk Hollow will exit at Gator Island.

FROM	TO			
	Frog Pond	**Skunk Hollow**	**Gator Island**	**Raccoon Corner**
Frog Pond	—	40	35	25
Skunk Hollow	37	—	39	24
Gator Island	42	29	—	29
Raccoon Corner	41	28	31	—

Table 2: The percentage of customers boarding (each station)

Personal Ethics Agreement Concerning University Assignments

Group Project

We submit this assignment and attest that we have applied all the appropriate rules of quotation and referencing in use at the University of Ottawa, http://web5.uottawa.ca/mcs-smc/academicintegrity/documents/2011/academic-integrity-students-guide.pdf. We attest that this work conforms to the regulations on academic integrity of the University of Ottawa. We understand that this assignment will not be accepted or graded if it is submitted without the signatures of all group members.

YILIN YANG — Name, Capital letters
300136515 — Student number

Yilin Yang — Signature
2020/12/5 — Date

FULIN ZHONG — Name, Capital letters
8338319 — Student number

Fulin Zhong — Signature
2020/2/5 — Date

DAVID PREM — Name, Capital letters
7795969 — Student number

David Prem — Signature
2020/2/5 — Date

HOSSEIN ZINAGHAJI — Name, Capital letters
300098309 — Student number

Signature
2020/2/5 — Date

SOROUSH SALEHI — Name, Capital letters
300140057 — Student number

Soroush Salehi — Signature
2020/02/05 — Date