CS6310 – Software Architecture & Design
Assignment #6 [150 points]: Mass Transit Simulation – Design Updates/Modifications (v6)
Fall Term 2018 – Prof. Mark Moss

Submission

- This assignment must be completed as an individual, not as part of a group.
- You must submit:
  - (1) A UML Class Diagram named **uml_class_diag.pdf**; and,
  - (2) A UML Sequence Diagram named **uml_seq_diagram.pdf**.
- You may submit multiple diagrams if needed – for example, if the diagrams are so large that dividing them makes it easier to present them in a more readable format. In that case, you may submit the multiple files with reasonable name extensions such as **uml_seq_diagram_1.pdf**, **uml_seq_diagram_2.pdf**, or **uml_seq_diagram_move.pdf**, **uml_seq_diagram_reset.pdf**, etc.
- Formats other than PDF must be pre-approved an Instructor or TA.

- You must notify us via a private post on Canvas and/or Piazza BEFORE the Due Date if you are encountering difficulty submitting your project. You will not be penalized for situations where Canvas is encountering significant technical problems. However, you must alert us before the due date – not well after the fact. You are responsible for submitting your answers on time in all other cases.
- Please consider that uploading files to Canvas might occasionally take a long time, even in the case of seemingly "relatively small" submissions. Plan accordingly, as submissions outside of the Canvas Availability Date will likely not be accepted. You are permitted to do unlimited submissions, thus we recommend you save, upload and submit often. You should use the same file naming standards for the (optional) "interim submissions" that you do for the final submission.

Scenario

Your requirement for this assignment involves updating your previous Unified Modeling Language (UML) based diagrams. You must provide updated versions of the UML Class Model and Sequence Diagrams to highlight the corrections and changes that you are proposing to address the new requirements covered below. Also, you must also supply short text descriptions to help clarify the intent, motivation and supporting details of your proposals.

Deliverables

For this assignment, you must review the new and updated requirements described below, and provide updated UML Diagrams that reflect how you would address the new requirements in your designs. These updated designs will be reviewed by your peers, and will also form the core of your later team-based implementation assignments. You must submit the following items:

1. Class Diagram – The earlier standards for your UML Class Model are still in effect (e.g. required elements, level of fidelity at least as detailed as the Udacity videos, etc.), and your diagram must address all previous and new requirements. You must also include a brief description highlighting the changes based on the new requirements introduced in this assignment. In other words, your

accompanying explanation does not have to cover the "core parts" of the UML diagrams from the earlier assignment – just the changes based on the new requirements.

2. Sequence Diagram – Similar to the UML Class Diagram, your Sequence Diagram must cover the updated "Move Next Bus" functionality to include passenger management and calculating system efficiency. As with the Class Diagram, you must also include a brief description highlighting the changes based on the new requirements introduced in this assignment.

Submission Requirements

- You must generate your diagrams using an automated tool (e.g. Argo UML, LucidCharts, Microsoft Visio, etc.) so that they are as clear & legible as possible. Even PowerPoint is allowed, though this is an excellent opportunity to use a tool that is more appropriately designed for UML as opposed to a general drawing tool like PowerPoint. The choice of tool(s) is yours; however, you should do a "sanity check" to make sure that your final diagrams are readable when exported to PDF; or, if necessary, some reasonable graphical format (PNG, JPG or GIF).

- Your design proposals will be evaluated based on a combination of factors, including:
    - The accuracy of your classes, along with the corresponding attributes and operations;
    - The correctness of the format of your design artifacts with respect to the UML Standards;
    - The clarity, conciseness and consistency of the text describing your proposals;
    - How thoroughly your proposals address the client's concerns; and,
    - The "general technical feasibility" of your proposals.

- You are not required to provide a working implementation of your proposals for this assignment. However, you must clearly describe the data that will be needed to support your proposals, and how and where that data will be stored and organized with respect to the various classes and other structures. Also, you should describe operations in enough detail to ensure that they can be reasonably implemented.

- With regards to the "general technical feasibility" comment: since you aren't actually developing a working prototype for this assignment, we aren't going to penalize you for incredibly specific technical issues. However, you also don't get to "wish away" or completely ignore the technical ramifications of your proposals. The design proposals that you submit for this assignment might very well become the designs that you are required to implement in later assignments, so you should ensure that your proposals are technically sound.

- You must designate which version of UML you will be using – either 1.4 (the latest ISO-accepted version) or 2.0 (the latest OMG-accepted version). There are significant differences between the versions, so your diagram must be consistent with the standard you've designated. Either version is acceptable at this point in the course. We might require a specific version for some of the other UML style components later in the course, especially considering the behavioral/activity-based diagrams; and, if so, we'll let you know.

- You are permitted to add a few sentences to explain any aspects of your design that you feel need extra clarification. These sentences are optional: non-submissions will not be penalized.

New Client Requests – The Areas of Focus for Your Design Proposals
The MTS System has been updated, and you should take some time to explore and analyze the new and improved capabilities of the application. However, even with the new capabilities, there's still much work to be done to satisfy the client's requests. First, we will describe the client's requests, which will be the focus of your efforts for this assignment. Your main goal will be to describe how you would modify the MTS application to address the client's requests, and then provide design artifacts that clearly, consistently and comprehensively capture the details of your proposed modifications. After we describe the client's requests, we will give an overview on the updated MTS system.

The client's requests for updates are reflected in four basic categories: ***Bus Changes***, ***Passenger Exchanges***, ***System Efficiency***, and ***Replays***.

- ***Bus Changes:*** When a simulation scenario is configured based on the input file, the buses are defined with a route, an initial speed and passenger capacity. The clients must be able to adjust the route, speed and/or passenger capacity of the bus during the simulation. Since this is a discrete-event simulation, any changes to the attributes of a bus will not take effect until the bus reaches its next stop. If the passenger capacity of a bus is reduced, then any "excess passengers" created by the change must also get off of the bus at the next stop. And if a route change is directed, then the client must be able to designate which stop of the new route that the bus will head to as its next destination.

- ***Passengers Exchanges:*** Currently, the system does not display passengers being picked up or dropped off by the buses in service. To improve the usefulness of the system, the impact of passengers being transported from their origin to their destination stops needs to be included. The clients would like to have the system model passengers arriving at, and departing from, each stop at certain frequencies. Similarly, passengers will also board (or leave) a bus when it arrives at a stop at certain frequencies. You must ensure that the simulation updates the number of passengers at each stop, and on the buses, during the simulation.

  Given the way the discrete-event simulation works, the system must handle the exchange of passengers at a particular stop with a specific sequence of steps. We will consider three groups of passengers at each stop: (1) the riders, (2) the waiting and (3) the transfers. When a bus arrives at a stop, that bus will be carrying a certain number of passengers - we will refer to this group as the "riders". Also, there will be a certain number of passengers already waiting at the stop for the next bus to arrive - we will refer to this group as "waiting". Finally, we will have a group called the "transfers", which always begins as an empty group when the bus first arrives. With these terms, the sequence of steps is:

  1. A certain number of new passengers will arrive at the stop to join the waiting group. The number of passengers who arrive at the stop will be determined using a uniform distribution

bounded by two (integer) values called *ridersArriveHigh* and *ridersArriveLow* (inclusively), with ridersArriveHigh ≥ ridersArriveLow. The size of the waiting group must be increased accordingly.

2. Before some of the passengers in the waiting group start boarding the bus that just arrived, they will allow the passengers already on the bus - the riders group - to get off. The number of passengers who get off the bus will be determined using a uniform distribution bounded by two values called *ridersOffHigh* and *ridersOffLow*, with ridersOffHigh ≥ ridersOffLow. The passengers who get off the bus should be added into the transfers group, and the size of riders group must be decreased accordingly. The transfers group is used to ensure that passengers who just got off the bus aren't selected to get right back onto the same bus.

3. Now, allow the passengers in the waiting group to board the bus. The number of passengers who get on the bus will be determined using a uniform distribution bounded by two values called *ridersOnHigh* and *ridersOnLow*, with ridersOnHigh ≥ ridersOnLow. The size of riders group must be increased accordingly, and the size of the waiting group must be decreased.

4. Finally, a certain number of passengers at the stop will decide to leave. Perhaps they've reached their destination, or are getting off to catch a connecting bus (goodness!); or, perhaps they feel that they've waited too long and have decided to take an alternative form of transportation (d'oh - bad!). In either case, the total number of passengers who leave the stop will be determined using a uniform distribution bounded by two values called *ridersDepartHigh* and *ridersDepartLow*, with ridersDepartHigh ≥ ridersDepartLow.

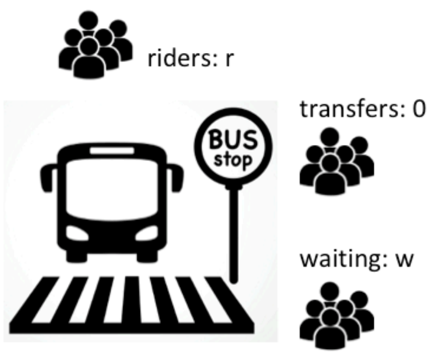
This final step is handled a bit differently than the previous steps, based on the sizes of the waiting and transfer groups. Suppose *ridersDepart* is the random number of passengers who've decided to leave the stop:

- If *ridersDepart* ≤ than the size of the transfers group, then that number of passengers from the transfers group have reached their destination. In this case, the size of the transfers group is reduced by the value of ridersDepart, and any remaining passengers in the transfers group are moved to the waiting group to catch their connecting bus.
- If *ridersDepart* > than the size of the transfers group, then all of the passengers in the transfers group have reached their destination and leave the stop. However, the remaining passengers that leave the stop are removed from the waiting group, and they depart unhappily - perhaps because they feel that the next bus has taken too long to arrive, etc.

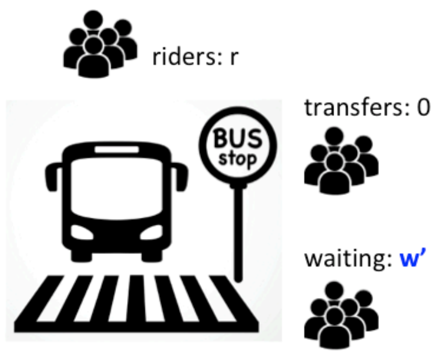
Note in both cases that the transfers group should be empty at the end of this step.

The waiting group can hold any number of passengers, but buses have a limited passenger capacity. The calculations above must be performed with respect to reasonable physical values. For example, if the current size of the riders group is 5, but 7 are selected to get off the bus, then only 5 will actually be selected - you shouldn't have a riders group with a size of -2 as a viable result. Similarly, if 10 passengers are selected from the waiting group to get on the bus where there are only 3 available seats, then only 3 passengers should be allowed to board the bus, and the others must remain in the waiting group.

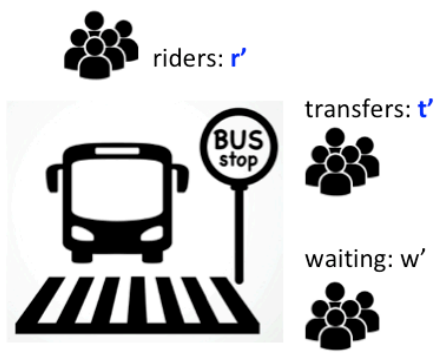
Here is a graphical representation of the passenger exchanges:



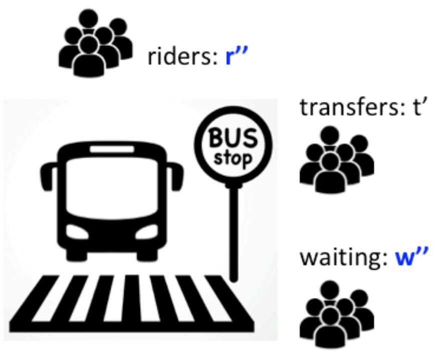
Step 0: The bus arrives



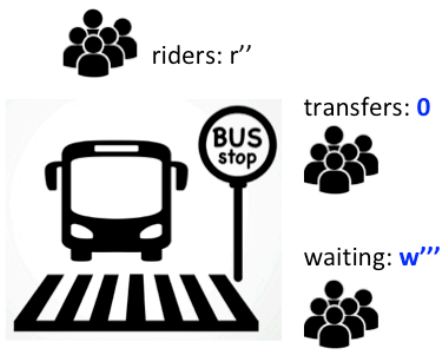
Step 1: New passengers arrive
ridersArrive = random value
w’ = w + *ridersArrive*



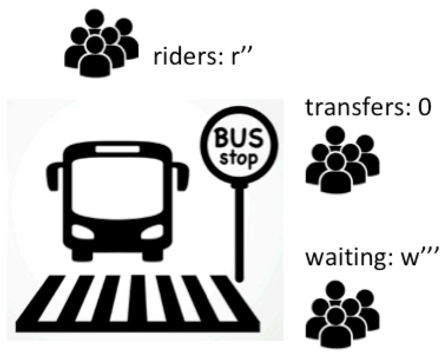
Step 2: Riders get off the bus
ridersOff = random value
r’ = r − *ridersOff* & **t’** = *ridersOff*



Step 3: Riders get on the bus
ridersOn = random value
r’’ = r’ + *ridersOn*
w’’ = w’ - *ridersOn*



Step 4: Passengers leave
ridersDepart = random value
w’’’ = w’’ + t’ − *ridersDepart*
t’’ = 0



Step 5: Passenger exchanges are complete
Don’t forget to calculate the next stop and travel time for the bus

- ***System Efficiency:*** The clients want to measure the effectiveness of their current configuration of buses, stops and routes for a scenario by measuring the how many passengers are waiting for a bus versus how many buses are in service, factoring in the speeds and capacities of the buses. The client’s goal is to minimize the number of waiting passengers while also minimizing the number of buses needed for support, and so they’re providing you with an objective function to measure the progress towards their goal. You must ensure that the system tracks and calculates these values so that the objective function and can be calculated and displayed at any given time.

  The objective function has two major components: (1) the number of waiting passengers, and (2) the cost of running the buses. The number of waiting passengers at a given point in time will be calculated as the sum of the passengers waiting at each of the different stops. A lower number of passengers waiting at stops will generally mean more passengers in transit towards their eventual destination, which is a significant part of the desired goal. The waiting passengers will be calculated as:

```
waiting_passengers() = SUM [passengers currently waiting at the stop]
for all stops in the system
```

The cost of running the buses will be measured as a function of the numbers and types of buses currently operating in the system. Buses with a faster speed and/or a higher capacity will be more expensive to operate, and will cost more from a utility standpoint. Being able to use fewer buses, and those buses not needing to travel as quickly and be as large (e.g. less fuel needed for operation), will be the other significant part of the goal. The bus costs will be calculated as:
bus_cost() = SUM [k_{speed} * bus_speed() + $k_{capacity}$ * bus_capacity()] for all buses in the system

Finally, the objective function will be expressed as:
system_efficiency() = $k_{waiting}$ * waiting_passengers() + k_{buses} * bus_cost() + $k_{combined}$ * waiting_passengers() * bus_cost()

All of the constants (**k_{speed}**, **$k_{waiting}$**, etc.) will be real/fractional values, and should be adjustable by the clients during the course of the simulation, but the system can initialize all of the constants with a value of one (1.0). You may display the system efficiency value as part of the normal graphical output, or provide a button (or similar method) for the client to request the current value.

- ***Replays:*** The clients appreciate the "Reset Buses" functionality of the current prototype, and would like to expand those capabilities. The clients would like to be able to "rewind" the simulation back a certain number of events in order to compare and contrast the effects of making different changes to the buses and other aspects of the simulation state. Your simulation must support the ability to undo the previous one through three events, and allow the clients to then resume normal operations from that point.

These modifications will improve the capabilities of the simulation system to model the real-world MARTA system, so that the clients can use it to help predict the effect of making changes more accurately.

Closing Comments & Suggestions
This is the information that has been provided by the customer so far. We (the OMSCS 6310 Team) will likely conduct an Office Hours where you will be permitted to ask us questions in order to clarify the client's intent, etc.

Quick Reminder on Collaborating with Others
Please use Piazza for your questions and/or comments, and post publicly whenever it is appropriate. If your questions or comments contain information that specifically provides an answer for some part of the assignment, then please make your post private first, and we (the OMSCS 6310 Team) will review it and decide if it is suitable to be shared with the larger class. Best of luck on to you this assignment, and please contact us if you have questions or concerns.