Assignment #8: Marta Transit Simulation - Implementation

CS6310 – Software Architecture & Design

Team 16

April 22, 2018

1. Client Requirements

Overall objective: improve current Marta simulation system to match real-life mass transportation situation. Four aspects of current MTS simulator need to be improved:

Riders

- A "normal" rider goes to a stop with the intent of catching a specific bus, or sequence of buses, that he/she knows will take them to a desired stop.
- System is capable to scale up to tens of thousands of riders on a daily basis. Consider application's overall performance.

Roads

- More accurately calculate the travel time between stops.
- Road has speed limit.
- Consider road traffic conditions.

Rails

- Train has lower operation frequency and two routes (N-S, E-W).
- Rail stations "intersect" with bus stops. Riders can take a combination of buses and trains.
- Train has no traffic condition impact.

Random

- This simulation allows users to make changes to the Marta system. The system can reflect the difference due to the changes.
- Consider refactoring.

2. Analysis, Assumptions and Design Explanation

Riders

In real world, riders at a stop wait for a bus/train on a desired route. To simulate this more accurately, we added a class RiderForRoute for the composition relationship between Route and Stop. This class stores rate information about riders getting on/off the specific route at the current stop (rateCatchingVehicle and rateLeavingVehicle). The rate information can be obtained from database. This class also handles rider exchange (exchangeRiders) between Route and Stop.

We added a collection of RiderForRoute objects, ridersAtStop, inside Stop class. The method getTotalWaiting inside Stop returns the total number of riders waiting at this stop, which will be called inside displayInternalStatus.

We are not creating a single object for each rider. Therefore, we do not need to worry that the number of riders becomes very large.

Roads

Instead of calculating Euclidean distance based on geological locations, we calculate travel times using two different methods for Bus and Train.

For travel time between buses, we added two methods getTravelTime and queryGoogleMap. During simulation, event ranking is converted to departure time and passed as a parameter to queryGoogleMap. Thus, the effect of traffic in real life is included.

For travel time between trains, we store average travel time from database inside a collection of travelTimes in RailStation class, The method getTravelTime will return the travel time stored in travelTimes if it is present and will queryGoogleMap if it is not present.

Rails

We consider rail routes as special bus routes and trains as special buses with large capacity. Therefore, it is not necessary to build separate classes like RailRoute or Train. We renamed BusRoute as Route, Bus as Vehicle, BusSytem as TransportationSystem for clarity. We added a boolean field isRail inside Route and Vehicle to indicate if the route is a rail route or a bus route, or the vehicle is a train or a bus.

Because trains are not affected by traffic and have constant travel time between stations, we build a class named RailStation. BusStop and RailStation are both subclasses of Stop. The method getTravelTime is different between BusStop and RailStation. RailStation has an additional field travelTimes.

When we import data from csv file, our assumption is that Bus Route has ID <= 1000 and Bus Stop has ID <= 1000000.

Random

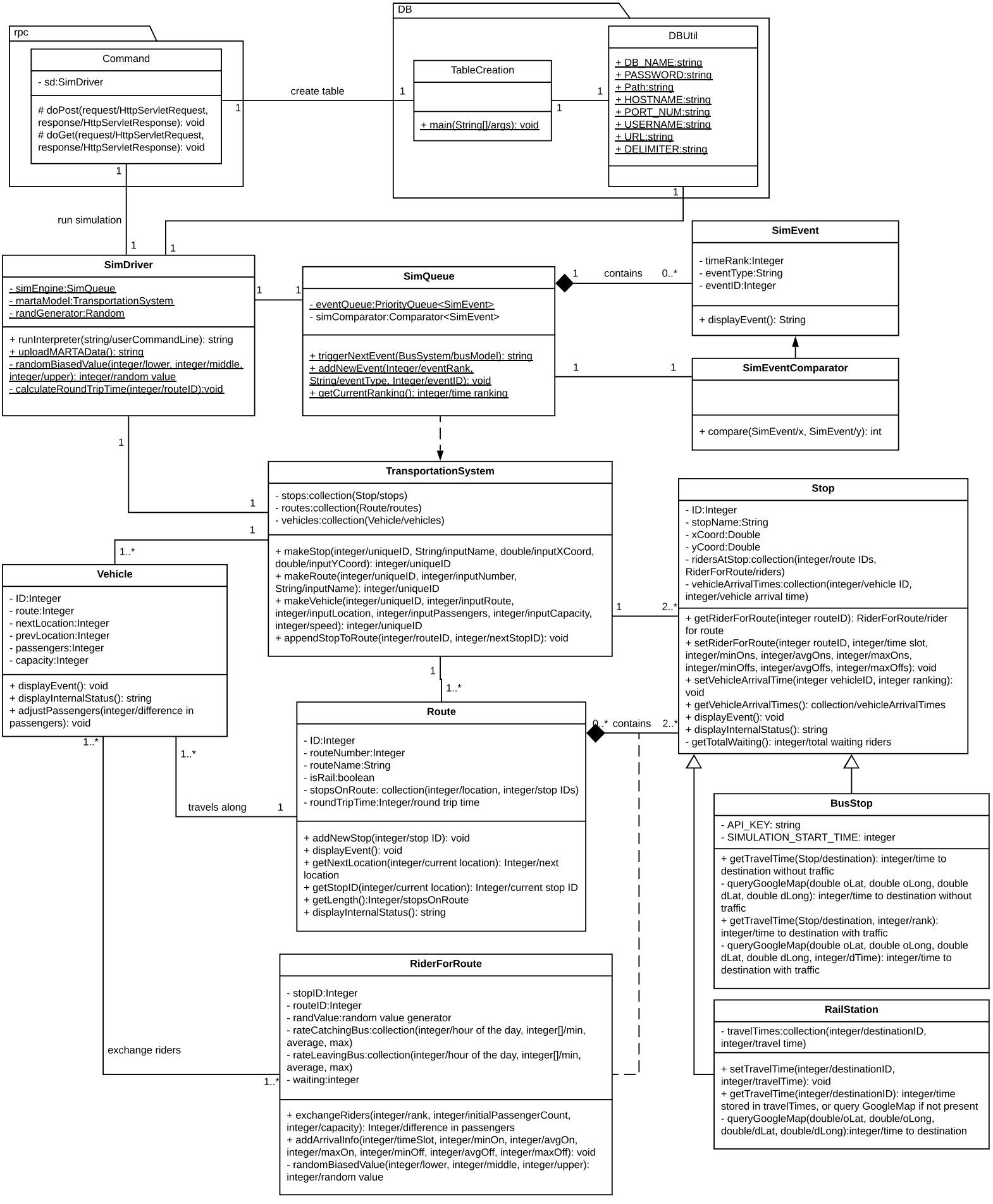
We add two new features.

The first new feature is that each route has a new field roundTripTime. This is an estimated round-trip travel time for a round and is calculated by adding up travel time between every stops. roundTripTime is calculated when uploading data from database or is updated after extending additional stops to a route.

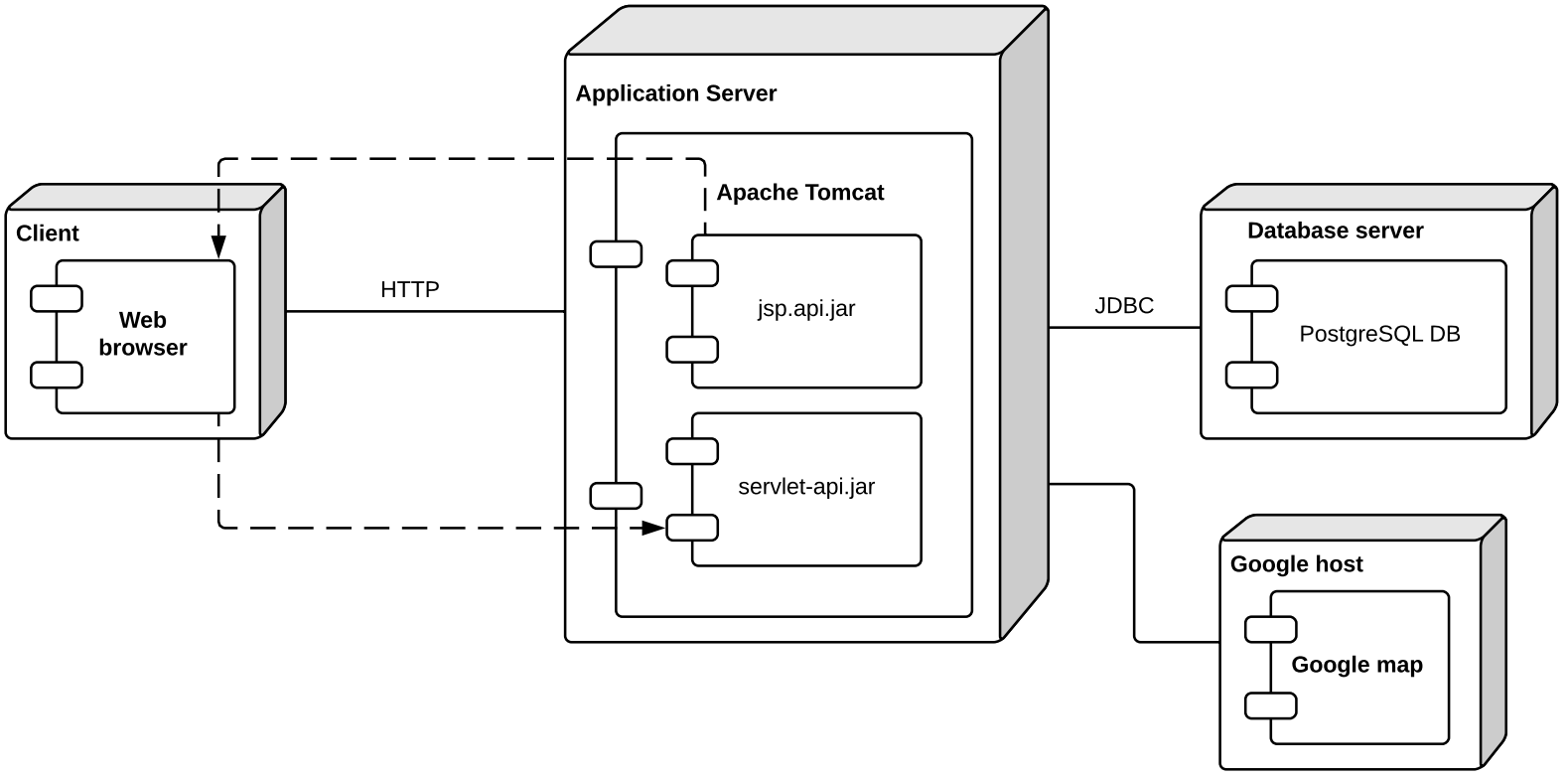
The second feature we add is that every stop can display next vehicle arrival times in system report. We add a collection of vehicleArrivalTimes in Stop class, which stores previous arrival times of vehicles at the stop. nextArrivalTime is calculated based on vehicle previous arrival time, route roundTripTime and current event time rank. The purpose of this modification is to make it easier for clients to decide if they want to add more buses or trains on a route.

3. Design Diagrams

Mass Transit Simulation - Class Diagram (UML 1.4)

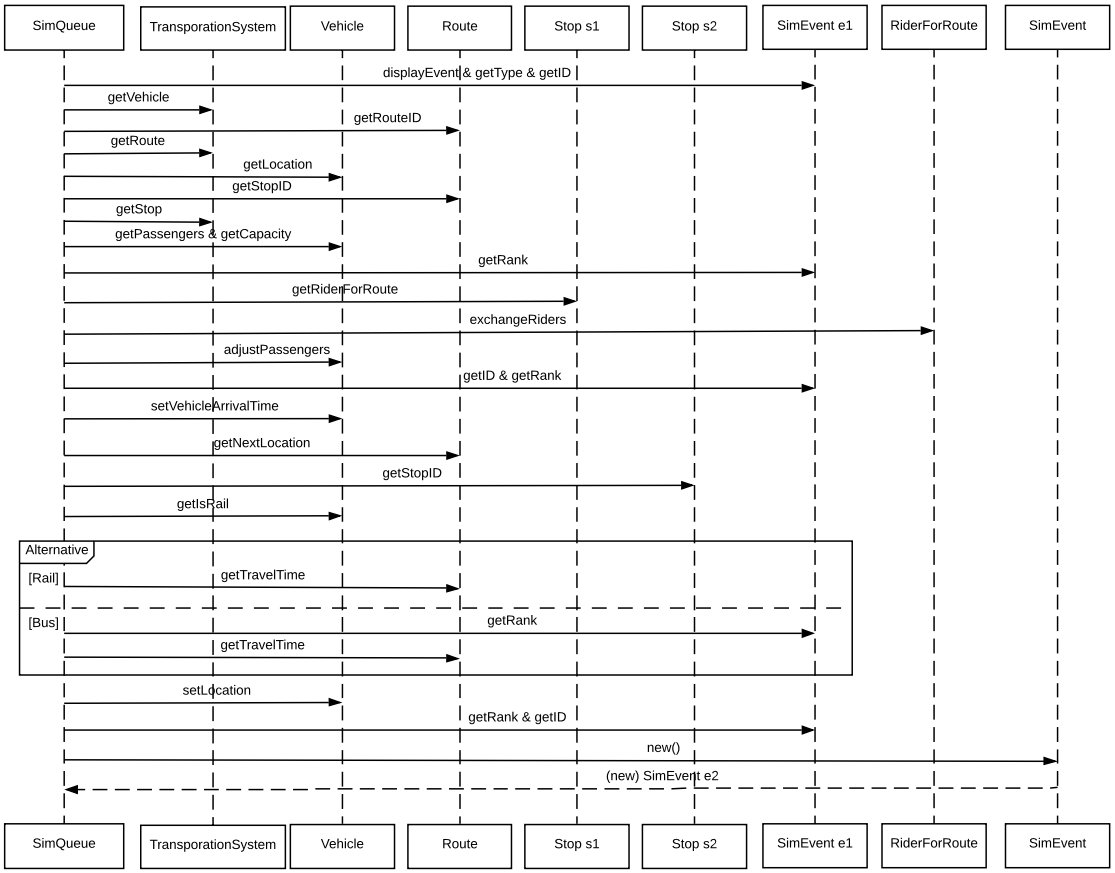


Mass Transit Simulation - Deployment Diagram (UML 1.4)

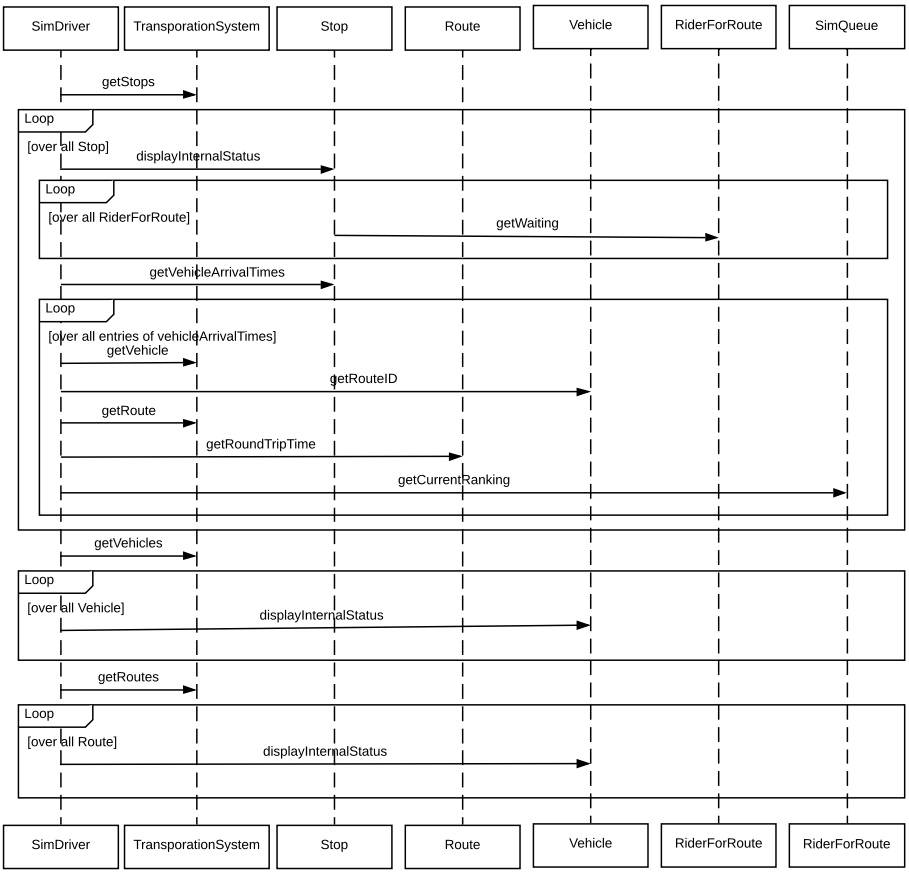


Mass Transit Simulation - Sequence Diagram (UML 1.4)

SimQueue: triggerNextEvent



SimDriver: system_report



Mass Transit Simulation - Use Case Diagram (UML 1.4)