CS 6310 Assignment 9

team 66

Document List:

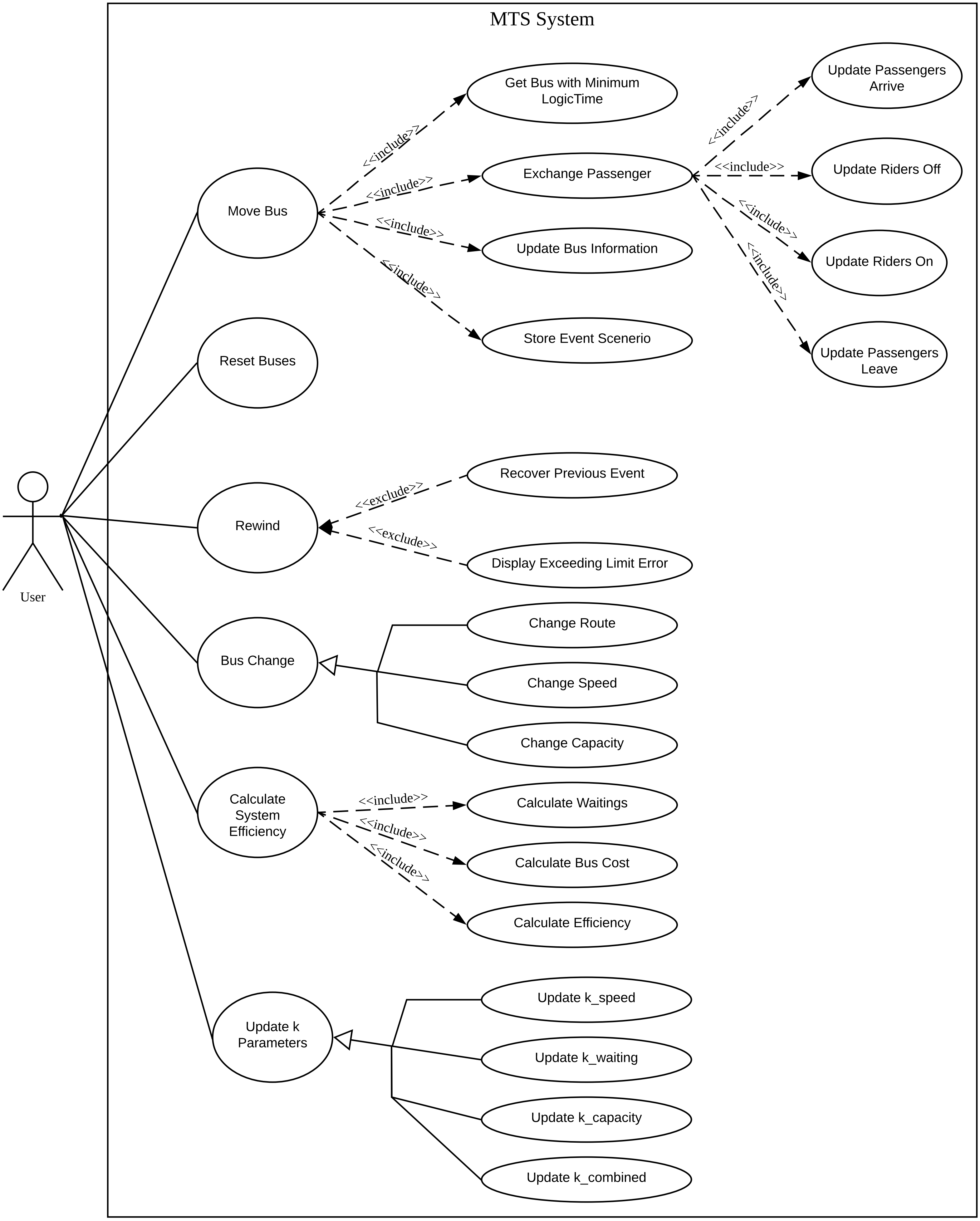
- MTS USE CASE DIAGRAM (UML 2.0)
- Deployment Diagram
- MTS CLASS DIAGRAM (UML 2.0)
- MTS SEQUENCE DIAGRAM_MOVEBUS
- MTS SEQUENCE DIAGRAM_REWIND
- MTS SEQUENCE DIAGRAM_EFFICIENCY
- SEQUENCE DIAGRAM_RESET AND BUSCHANGE
- MTS Design Document

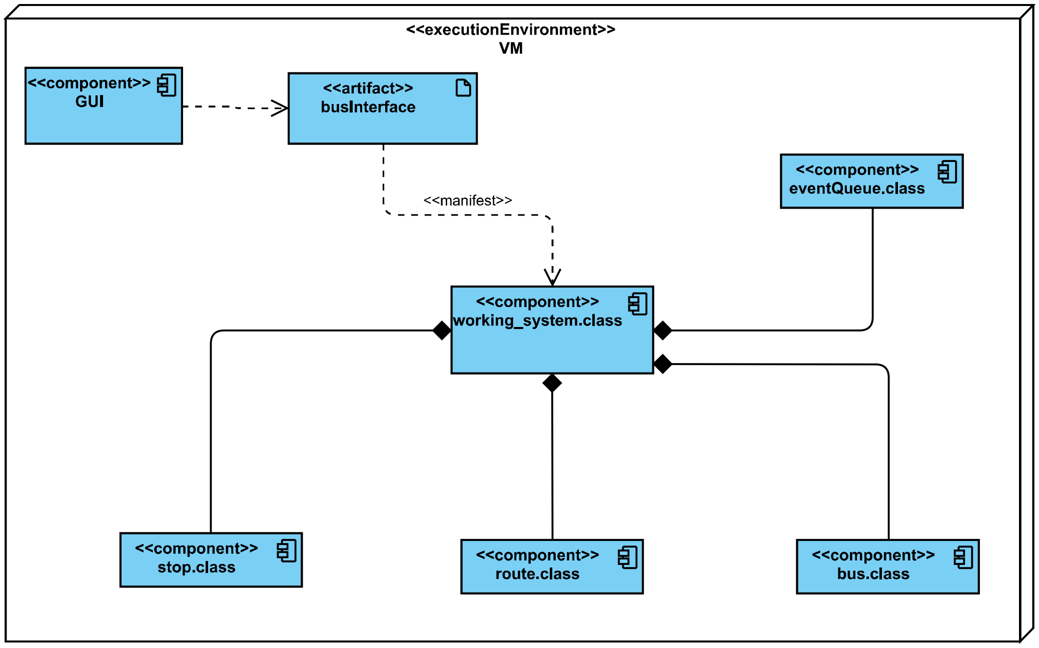
Team Member:

- Lingqiong Guo
- Menglu Lin
- Min Ni
- Liu Su
- Linyue Zhao

A9 MTS USE CASE DIAGRAM (UML 2.0)

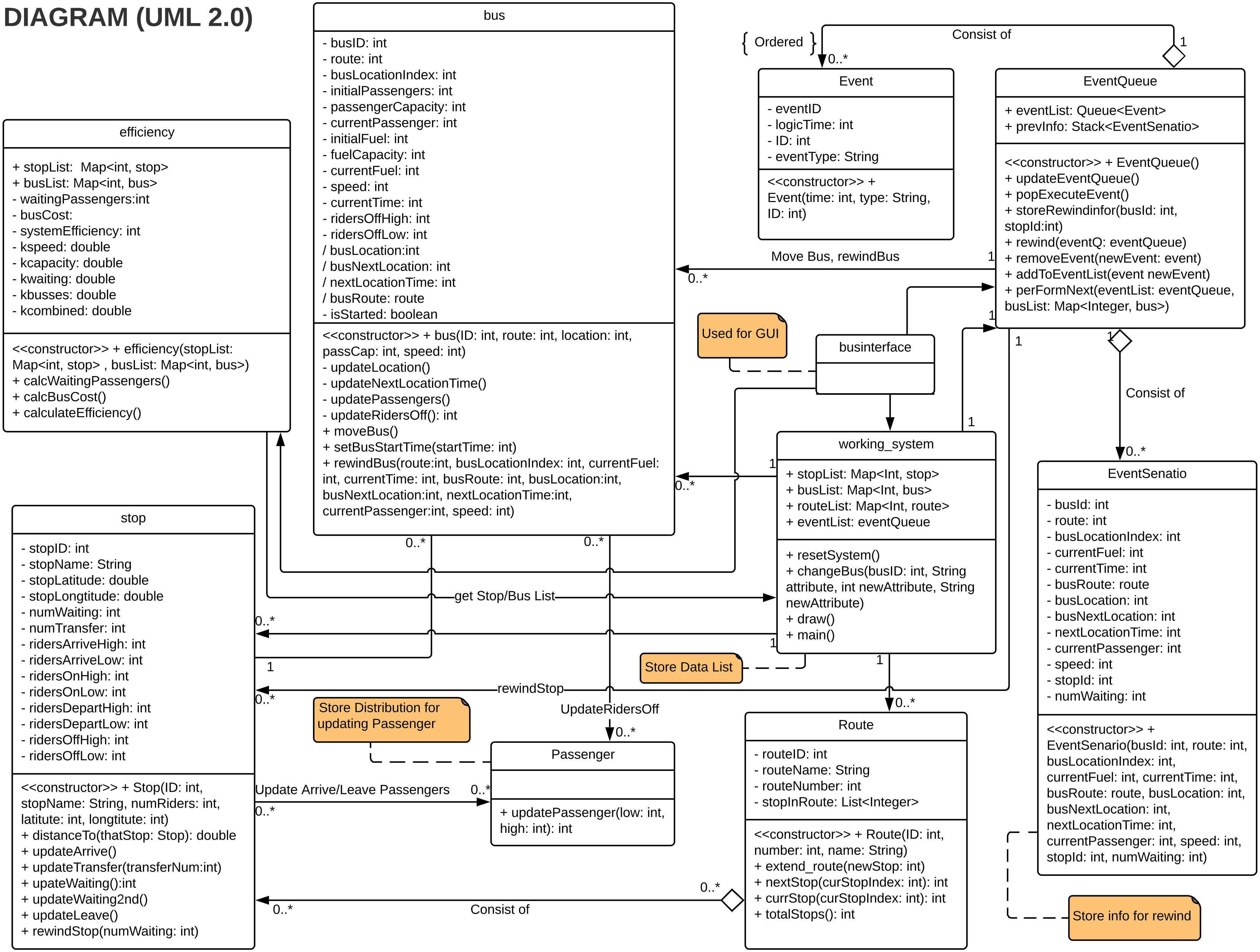
Group 66 | December 2, 2018




<<executionEnvironment>>
VM
<<component>>
GUI
<<artifact>>
busInterface
<<manifest>>
<<component>>
eventQueue.class
<<component>>
working_system.class
<<component>>
stop.class
<<component>>
route.class
<<component>>
bus.class

MTS CLASS DIAGRAM (UML 2.0)

Group 66 | December 2, 2018



bus

- busID: int
- route: int
- busLocationIndex: int
- initialPassengers: int
- passengerCapacity: int
- currentPassenger: int
- initialFuel: int
- fuelCapacity: int
- currentFuel: int
- speed: int
- currentTime: int
- ridersOffHigh: int
- ridersOffLow: int

/ busLocation:int
/ busNextLocation: int
/ nextLocationTime: int
/ busRoute: route

- isStarted: boolean

<<constructor>> + bus(ID: int, route: int, location: int, passCap: int, speed: int)

- updateLocation()
- updateNextLocationTime()
- updatePassengers()
- updateRidersOff(): int

+ moveBus()
+ setBusStartTime(startTime: int)
+ rewindBus(route:int, busLocationIndex: int, currentFuel: int, currentTime: int, busRoute: int, busLocation:int, busNextLocation:int, nextLocationTime:int, currentPassenger:int, speed: int)

efficiency

+ stopList: Map<int, stop>
+ busList: Map<int, bus>

- waitingPassengers:int
- busCost:
- systemEfficiency: int
- kspeed: double
- kcapacity: double
- kwaiting: double
- kbusses: double
- kcombined: double

<<constructor>> + efficiency(stopList: Map<int, stop> , busList: Map<int, bus>)

+ calcWaitingPassengers()
+ calcBusCost()
+ calculateEfficiency()

Event

- eventID
- logicTime: int
- ID: int
- eventType: String

<<constructor>> + Event(time: int, type: String, ID: int)

EventQueue

+ eventList: Queue<Event>
+ prevInfo: Stack<EventSenatio>

<<constructor>> + EventQueue()

+ updateEventQueue()
+ popExecuteEvent()
+ storeRewindinfor(busId: int, stopId:int)
+ rewind(eventQ: eventQueue)
+ removeEvent(newEvent: event)
+ addToEventList(event newEvent)
+ perFormNext(eventList: eventQueue, busList: Map<Integer, bus>)

businterface

working_system

+ stopList: Map<Int, stop>
+ busList: Map<Int, bus>
+ routeList: Map<Int, route>
+ eventList: eventQueue

+ resetSystem()
+ changeBus(busID: int, String attribute, int newAttribute, String newAttribute)
+ draw()
+ main()

EventSenatio

- busId: int
- route: int
- busLocationIndex: int
- currentFuel: int
- currentTime: int
- busRoute: route
- busLocation: int
- busNextLocation: int
- nextLocationTime: int
- currentPassenger: int
- speed: int
- stopId: int
- numWaiting: int

<<constructor>> + EventSenario(busId: int, route: int, busLocationIndex: int, currentFuel: int, currentTime: int, busRoute: route, busLocation: int, busNextLocation: int, nextLocationTime: int, currentPassenger: int, speed: int, stopId: int, numWaiting: int)

stop

- stopID: int
- stopName: String
- stopLatitude: double
- stopLongtitude: double
- numWaiting: int
- numTransfer: int
- ridersArriveHigh: int
- ridersArriveLow: int
- ridersOnHigh: int
- ridersOnLow: int
- ridersDepartHigh: int
- ridersDepartLow: int
- ridersOffHigh: int
- ridersOffLow: int

<<constructor>> + Stop(ID: int, stopName: String, numRiders: int, latitute: int, longtitute: int)

+ distanceTo(thatStop: Stop): double
+ updateArrive()
+ updateTransfer(transferNum:int)
+ upateWaiting():int
+ updateWaiting2nd()
+ updateLeave()
+ rewindStop(numWaiting: int)

Passenger

+ updatePassenger(low: int, high: int): int

Route

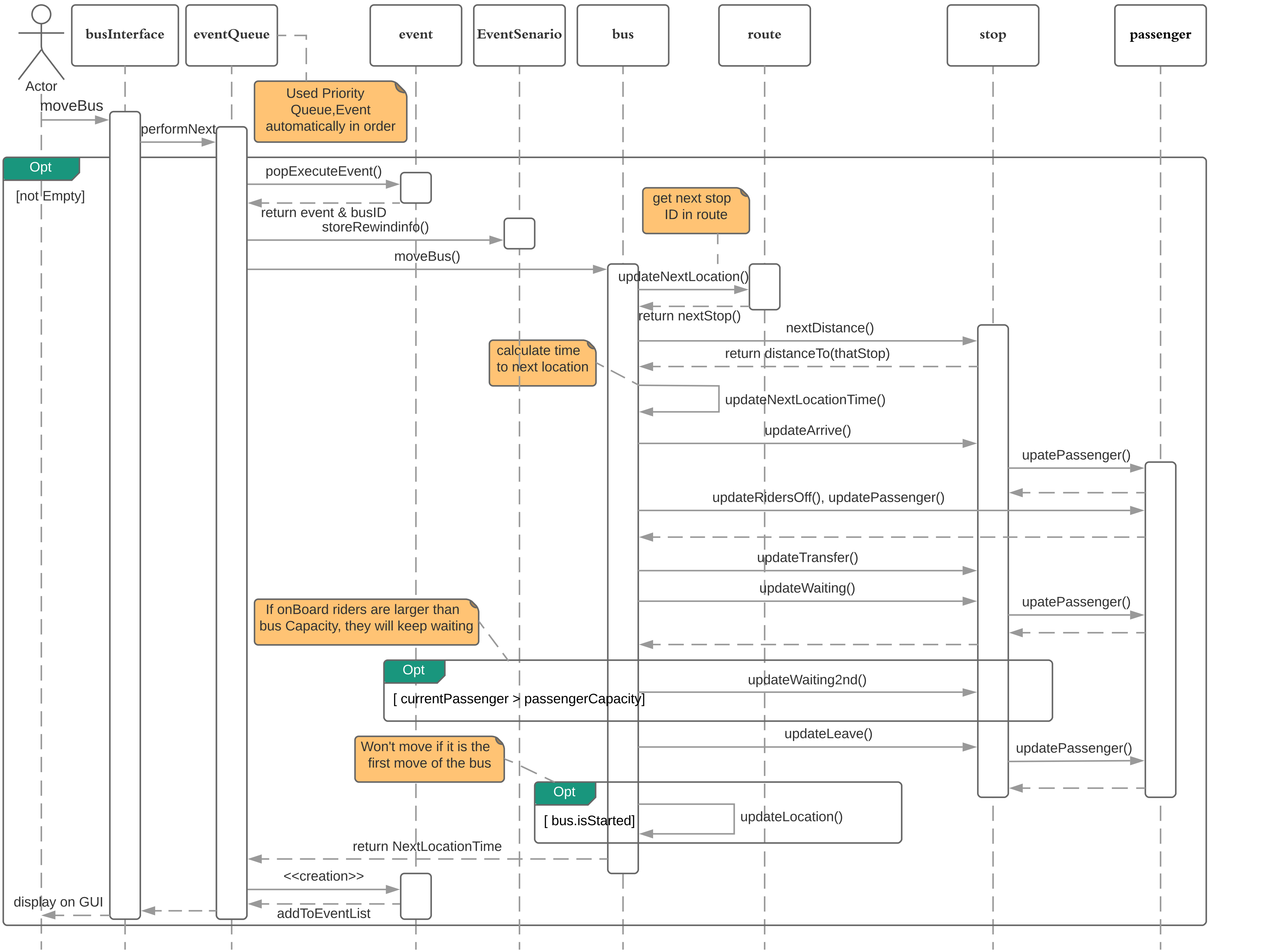
- routeID: int
- routeName: String
- routeNumber: int
- stopInRoute: List<Integer>

<<constructor>> + Route(ID: int, number: int, name: String)

+ extend_route(newStop: int)
+ nextStop(curStopIndex: int): int
+ currStop(curStopIndex: int): int
+ totalStops(): int

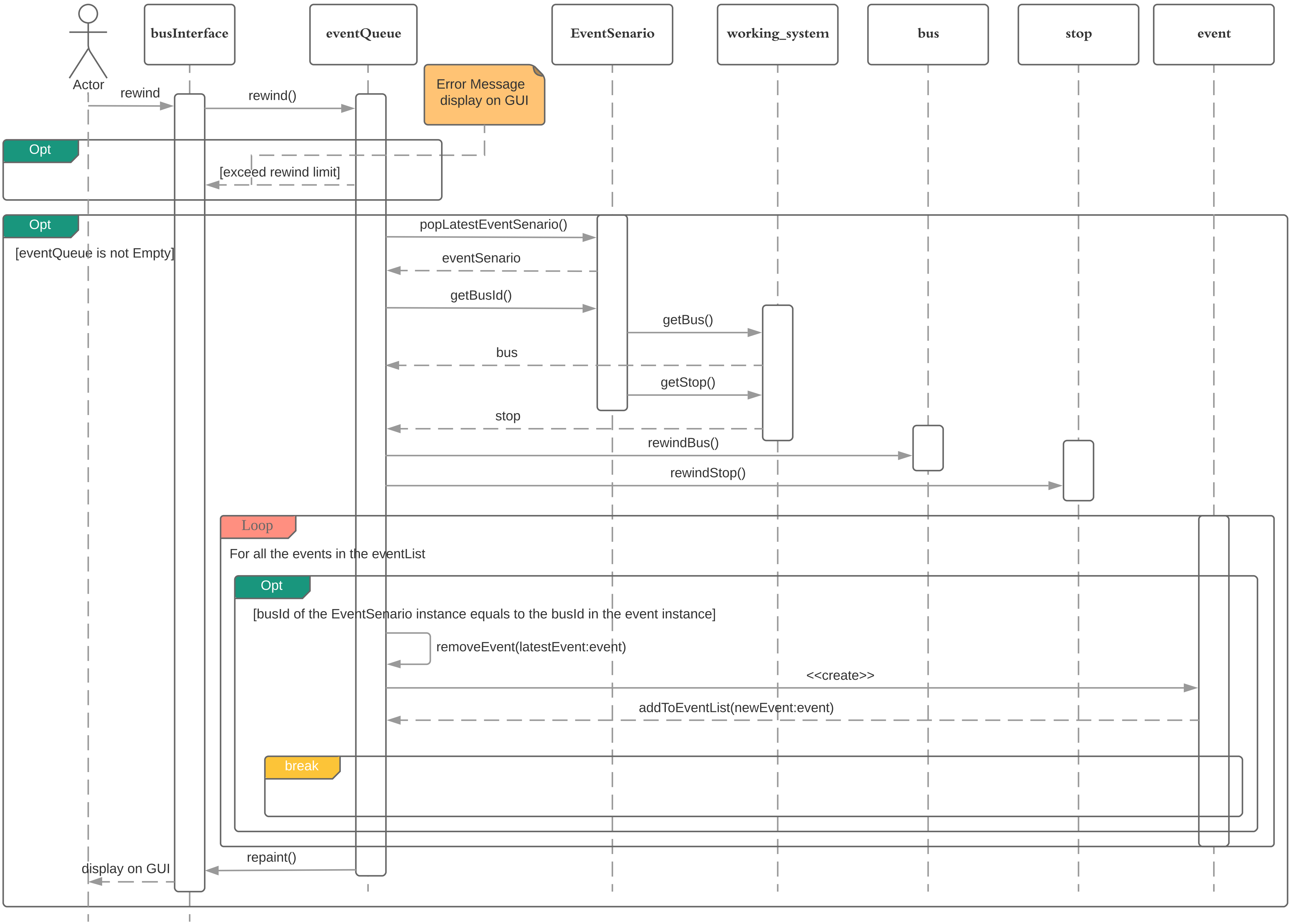
A9 MTS SEQUENCE DIAGRAM_MOVEBUS (UML

Group 66 | December 2, 2018



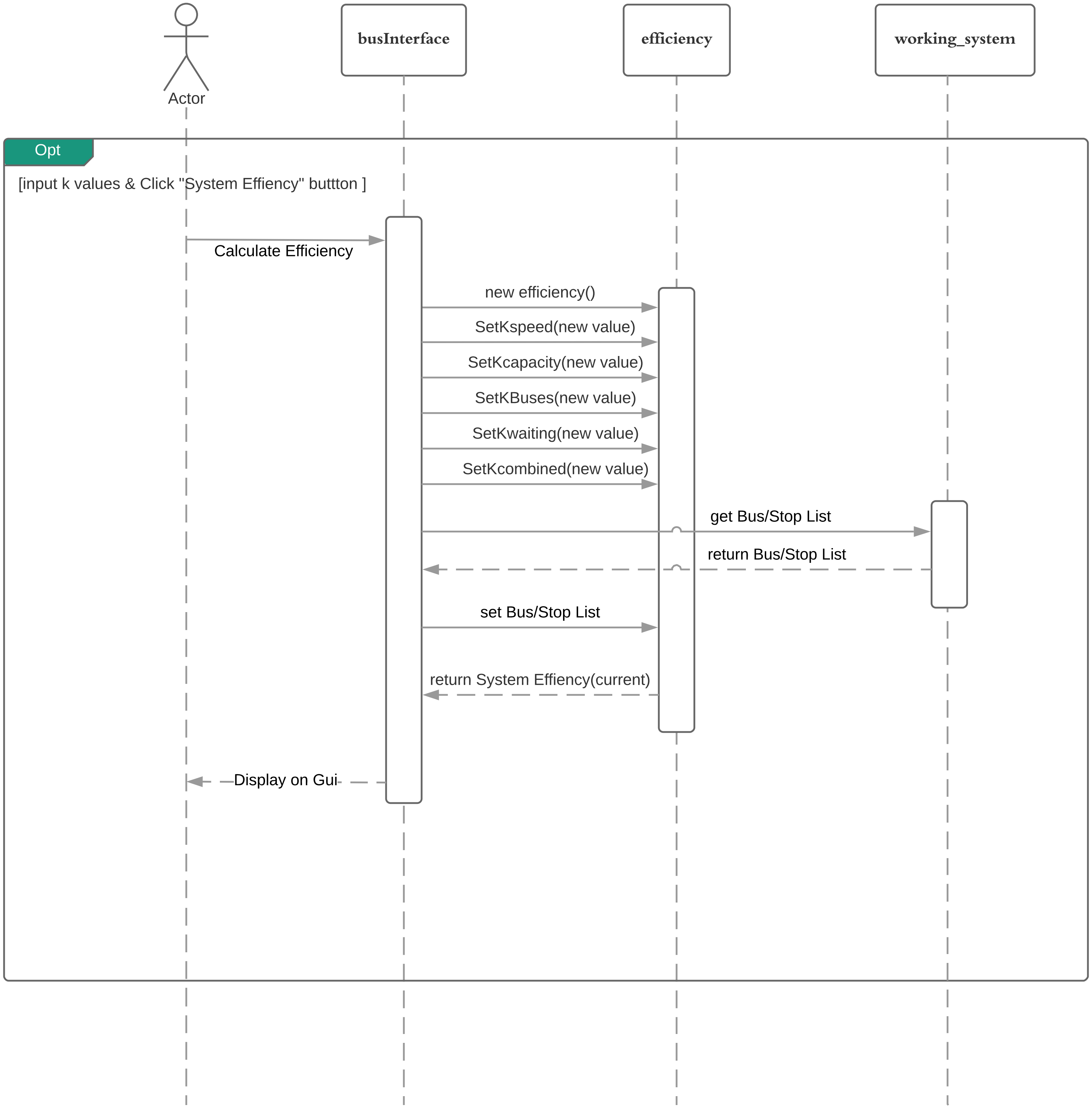
A9 MTS SEQUENCE DIAGRAM_REWIND (UML 2.0)

Group 66 | December 2, 2018



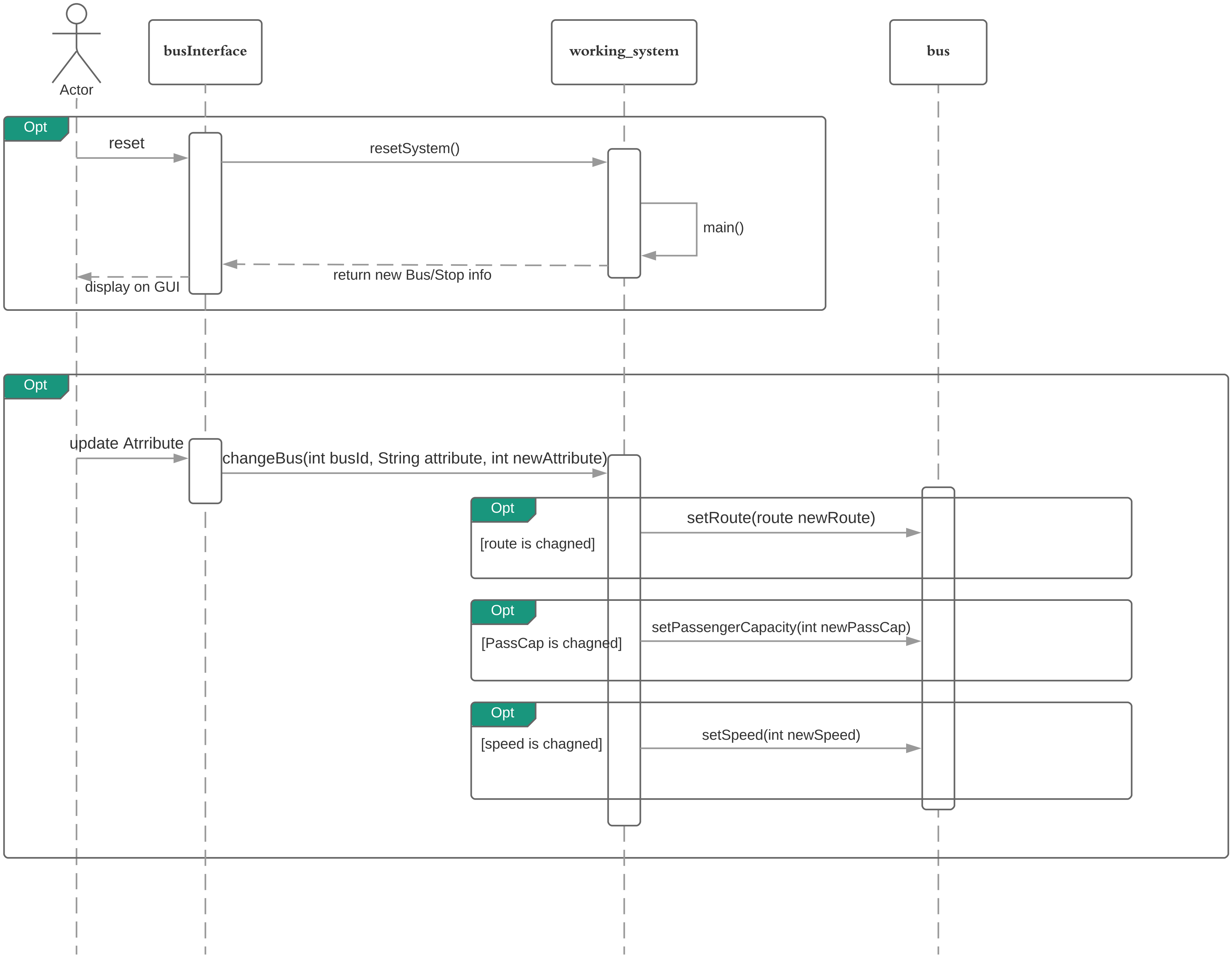
A9 MTS SEQUENCE DIAGRAM_EFFICIENCY (UML 2.0)

Group 66 | December 2, 2018



A9 SEQUENCE DIAGRAM_RESET AND BUSCHANGE (UML 2.0)

Group 66 | December 2, 2018



MTS Design Document

Group 66

1. Introduction

This application is a Mass Transit Simulation (MTS) system. The prototype application that has been provided simulates the interaction of buses moving along a route of stops and allowing passengers to board and later depart the bus at different stops.

2. Features and Assumptions

2.1. Move Bus Along the Route

2.1.1. Features

This system corresponds to transitioning a bus from its current stop to its next stop based on the distance between the two stops, and the current speed of the bus. The arrival of the bus at its next location also triggers actions involving current passengers getting off the bus, followed by new passengers getting back onto the bus.

2.1.2. Assumptions

The distance and speed calculations are adjusted with conversion factors so that the rank of an event corresponds to when the event would be executed in minutes from the start of the simulation run. We have the following assumptions when calculating arrival time:

- There is relatively straight path (e.g. sequence of roads) between the two stops so any road curvature is insignificant.
- We use Euclid distance to calculate the distance: distance [d/m/s] = square_root ((x1 – x2)2 + (y1 – y2)2).

2.2. Passenger Exchange

2.2.1. Features

Passengers will represent the people who ride buses from their arrival stop to their destination stop. We will simulate the arrival of passengers at each stop using a random distribution based on the MARTA data that we’ve collected as part of the SLS program. The data is contained in a CSV file and is used to generate upper and lower limits for the probability distributions of passenger arrivals and departures for different hours during the day.

2.2.2. Assumptions

The distribution for updating passengers each time was assumed to be uniform between the given upper bound and lower bound. A csv file will include the probability distribution limits for passengers arriving at and departing from stops and getting on and off of buses. This file format will consist of one line for each stop, and each line will include the probability parameters that stop in the following order:

<Stop ID>, <Riders Arrive High>, <Riders Arrive Low>, <Riders Off High>, <Riders Off Low>, <Riders On High>, <Riders On Low>, <Riders Depart High>, <Riders Depart Low>

The higher and lower bound to update passengers on the bus and at the stop are as follows:

- Riders off the bus High: distribution csv file Colum B
- Riders off the bus Low: distribution csv file Colum C
- Riders arrived at the stop High: distribution csv file Colum D
- Riders arrived at the stop Low: distribution csv file Colum E
- Riders go onboard High: distribution csv file Colum F
- Riders go onboard Low: distribution csv file Colum G
- Riders left the stop High: distribution csv file Colum H
- Riders left the stop Low: distribution csv file Colum I

2.3. Bus Change

2.3.1. Features

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 route, speed and/or passenger capacity of the certain bus should be able to adjust by the clients during the simulation.

2.3.2. Assumptions

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 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.

2.4. System Efficiency Calculation

2.4.1. Features

The simulation should allow the clients 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 system tracks and calculates these values so that the objective function and can be calculated and displayed at any given time.

2.4.2. Assumptions

All the constants (**`kspeed`**, **`kwaiting`**, etc.) will be real/fractional values, and should be adjustable
by the clients during the simulation, but the system initializes all the constants with a value of one (1.0). A button is provided for the client to request the current value.

2.5. Reset and Replay

2.5.1. Features

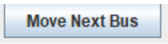
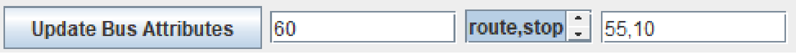
The “Reset Buses” functionality of the current prototype will be provided and would like to expand those reset capabilities. System also allows to “rewind” the simulation back a certain number of events to compare the effects of making different changes to the buses and other aspects of the simulation state.

2.5.2. Assumptions

Simulation only supports the ability to undo the previous one through three events and allow the clients to then resume normal operations from that point.

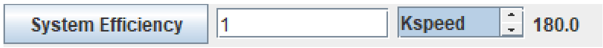
3. Software Instructions

Instruction steps:

- **Launching the system**:
  Open the software to trigger the GUI to run.
- **Loading the txt file and csv file**：
  txt file: event file
  csv file: probability distribution file
- **Move next bus**:
  Click the  to trigger next event
- **Reset bus**:
  Click the to trigger this event, then all the bus states will be reset as initials.
- **Update Bus Attributes**:
  

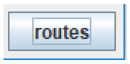
  60 route,stop 55,10
  Fill the [BusID] blank to tell the system which bus should be updated. Choose the attributes select button to tell system which bus attribute (“route,stop”, “passenger capacity”, “speed”) is changed. Fill the [setNewAttribute] blank what value should it be changed to. Here the new route id and stop id are seperated by comma.
  Then click “Update Bus Attributes” button to trigger this event.
- **Rewind**:
  Click the button to rewind to last event state.

- **System Efficiency**:

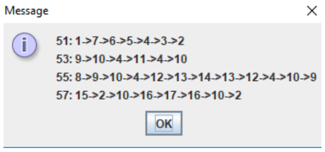


Select the 5 values select button (Kspeed, Kbuses, Kstop, Kpassenger, Kcombined) to tell system which k value need change and fill the blank [1] to tell system what new value the corresponding attribute will be changed to. Then click “System Effciency” button to display the newest efficiency value.

- Display Route list and Stop list



Click this button will display all the route list I in the system and the corresponding stop list.



```
51: 1->7->6->5->4->3->2
53: 9->10->4->11->4->10
55: 8->9->10->4->12->13->14->13->12->4->10->9
57: 15->2->10->16->17->16->10->2
```