Truck Platooning

1st Tural Hasanov
Fachhochschule Dortmund
Dortmund, Germany
tural.hasanov001@stud.fh-dortmund.de

2nd Rahim Hashimov
Fachhochschule Dortmund
Dortmund Germany
rahim.hashimov001@stud.fh-dortmund.de

3rd Seda Sensoy
Fachhochschule Dortmund
Dortmund Germany
seda.sensoy003@stud.fh-dortmund.de

4th Veli Ates
Fachhochschule Dortmund
Dortmund Germany
veli.ates001@stud.fh-dortmund.de

I. Abstract – Seda Sensoy

Platooning technology has advanced significantly in the recent decade, but to take the next step toward truck platooning implementation. The goal of this document is to illustrate the platoon architecture and use cases that will serve as the foundation for future definition and implementation of the services. The simulations are performed using the Java programming language. The high-level use cases for platoon are described in detail. First, the technique for generating use cases is specified, and then the high level use cases for platoon and extra one-lines / assumptions are added, which may be combined to generate comprehensive use cases as part of the technical deliverables.

II. Motivation – Seda Sensoy

Platooning is the coordinated movement of two or more vehicles in a convoy. The spacing between cars in a platoon may be closely managed with no additional driver fatigue. Platooning cars may also respond swiftly to the leader's braking actions, far faster than the usual human driver's response time. As a result, vehicles in a platoon can follow each other far closer than what is normally considered a safe following distance under human operation. It is inferred that platooning technology must be exceedingly resilient under very close following conditions before it can be used on a large scale. The three key reasons for platooning are decreased greenhouse gas emissions, lower operational costs, and the development of self-driving and V2V transportation systems for safer streets. Our project's major purpose is to pave the road for the deployment of truck platooning to optimize fuel usage, traffic safety, and throughput.

Following requirements are defined:

- The vehicles must be compatible to ensure correct and safe operation.
- Vehicles should safely communicate with vehicles and infrastructure.
- Communication should be secure.
- The components of the system should process multiple tasks simultaneously.
- The system should reduce the fuel usage.
- Each platoon member must be capable of performing all platoon roles (leading, following, and trailing).
- All platoon members must be equipped with wireless V2V communication and ambient awareness sensors.

III. Sketch of Approach – Rahim Hashimov

Here's a sketch of an approach for truck platooning (Fig. 1.):

1. Design platoon formation: Determine the desired formation for the trucks in the platoon, considering factors such as safety, fuel efficiency, and road conditions. This may involve using advanced algorithms to optimize the formation and may require real-time adjustments based on changing road and traffic conditions.
2. Implement communication system: Develop a distributed communication system that allows the trucks in the platoon to exchange information such as speed, acceleration, and position. This communication system should be reliable and low latency to enable real-time coordination and control.
3. Develop vehicle control system: Implement a vehicle control system that uses data from the communication system to adjust the speed, acceleration, and braking of the trucks in the platoon. This control system should be designed to ensure safety and efficiency and should take into account factors such as road and traffic conditions, as well as the performance characteristics of the individual trucks in the platoon.
4. Code and Simulate: Test the platooning system in a controlled environment, such as a closed test track, and validate its performance and safety. This may involve running simulations, testing individual components, and gradually scaling up to full-scale tests with multiple trucks in the platoon.

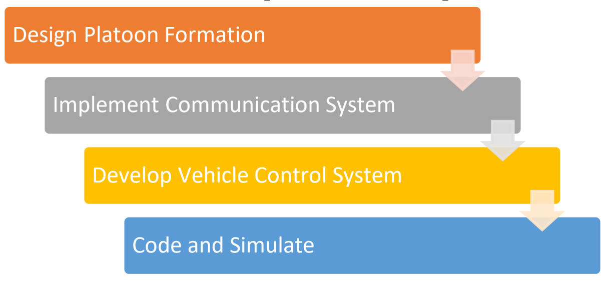

Design Platoon Formation
Implement Communication System
Develop Vehicle Control System
Code and Simulate

Fig. 1. Sketch of approach

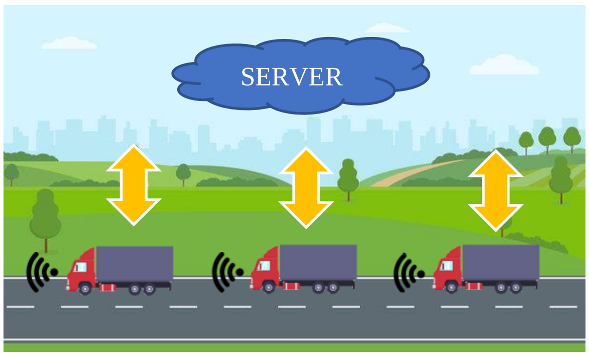


Fig. 2.

IV. Concept

A. Architecture – Veli Ates

To design an appropriate distributed and parallel architecture for truck platooning, the following requirements and characteristics must be fulfilled:

- Scalability: The architecture should be able to handle a large number of trucks and scale up as the fleet grows.
- Real-time communication: The platooning system requires real-time communication between the trucks, which should be low latency and reliable.
- Fault tolerance: The architecture should be resilient to hardware and software failures and be able to continue operating in the event of a fault.
- Security: The platooning system must be secure and tamper-proof, to prevent unauthorized access or malicious attacks.
- Data processing: The system should be able to process large amounts of data from sensors and other sources to support decision-making and control of the platoon.

There are several communication technologies and protocols that can be used to achieve distributed communication in truck platooning, including:

- Dedicated Short Range Communication (DSRC): DSRC is a wireless transmission protocol developed for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication that runs in the 5.9 GHz range. DSRC allows for high-speed, low-latency communication between trucks in a platoon, as well as the sharing of data such as vehicle speed, location, and acceleration.
- Cellular Vehicle-to-Everything (C-V2X): C-V2X is a cellular communication protocol that allows vehicle-to-vehicle (V2V), vehicle-to-internetwork (V2I), and vehicle-to-network (V2N) connection. C-V2X communicates between trucks in a platoon using cellular networks to enable high-speed, low-latency communication and may be used to transmit data such as traffic information, weather data, and road conditions.
- WiFi: WiFi can be used to communicate amongst trucks in a platoon, especially when DSRC or C-V2X are not available. WiFi is capable of high-speed, low-latency connectivity and may be used to share data such as video feeds from the trucks' cameras and other sensor data.

Overall, designing an effective distributed and parallel architecture for truck platooning requires a comprehensive understanding of the technical requirements and operational characteristics of the platooning system, as well as expertise in software engineering, data science, and cloud computing.

B. Use Cases – Veli Ates

In this section the methodology to derive a context from different scenarios will be discussed. Use-cases define how a system should respond to interactions from the system's user or surroundings, such as other traffic participants or road conditions, under various scenarios. A use case in our project is a complete scenario that includes motions, operations, and events.

The main high level use cases of platoon are:

- Platoon Formation
- Engaging to Platoon (join from behind by existing platoon)
- Platooning
    - Steady State Platooning
    - Emergency Breaking
    - Gap Adaptation
- Disengage Platoon (leave)

Platoon Formation

A platoon enabler service activates at least two different vehicles. The platoon enabler service will propose that the two vehicles form a platoon based on some business logic.

1. The platoon enabler (offboard service) matches some vehicles by sending a request to form a platoon together.
2. The platoon enabler assigns first vehicle as leader, and the others as following trucks.
3. Vehicles activates platooning (communicates).
4. The platoon formation finished.

Engaging to Platoon

The Platoon Candidate (single vehicle or existing platoon) sends an engagement request to the platoon target (leader vehicle or existing platoon) in front via wireless communication (V2V). The candidate joins the platoon after receiving approval from the leader vehicle.

1. The ego vehicle sends a request to join to the target vehicle.
2. The platoon system evaluates the join request and based on the decision, it sends acceptance or rejection response to candidate vehicle.
3. If platoon accepts candidate vehicle, then the candidate starts V2V communication between platoon members
    1. The join event is reported to other platoon members.
    2. The ego vehicle closes the distance to the platooning distance.
    3. If needed the platoon reduces speed to enable the ego vehicle to join the platoon.
    4. Once the distance gap is decreased the platoon is in steady state platooning
4. If platoon rejects candidate vehicle, then the communication between candidate vehicle and platoon will be terminated.

Platooning

Steady State Platooning

A line of two or more autonomous cooperative vehicles maintains a tight distance by employing wireless communication (V2V). To achieve the identical behavior in all platoon members, the speed, coordinates, wheel angle, and other attributes are continually measured and transmitted to all platoon members.

1. The leader vehicle is broadcasting information on V2V for the other platoon members to consume.
2. Based on V2V, sensor, and vehicle internal information, the ego vehicle regulates safe distance and speed.

Emergency Breaking

The ego truck is a member of a platoon, and any of the trucks ahead of it (in the platoon) performs Emergency Braking. To avoid a collision, the ego truck applies emergency braking and safely decelerates. Other trucks in the platoon behind the ego vehicle perform Emergency Braking and safely decelerate.

1. Any platoon driving state.
2. Emergency Braking trigger detected.
3. Target truck broadcasts its intended and actual deceleration immediately via V2V.
4. The leader truck evaluates a possible collision via V2V then the leader truck transmits that via V2V.
5. The Platoon is automatically split if more trucks precede the target vehicle that initiates the emergency braking event. The rest of the Platoon will continue to drive.
6. Executes emergency braking until the emergency is no longer present or the target vehicle has stopped completely.

Gap Adaptation

A platoon is running in steady state platooning. I2V communication informs all members in the platoon to increase distances between trucks for a specific zone defined by GPS start and end points. It could also be a specific speed, lateral positioning, etc...

1. Any platoon driving state.
2. As they drive in front of an I2V, all trucks in the platoon receive instructions from the I2V to slow down due to specific traffic, including a specific vehicle gap.
3. Internally, the platoon coordinates to account for changes in speed, inter-vehicle distance, and so on.
4. The platoon resumes its previous configuration in a coordinated fashion at the end of the adaptation section.

Disengage Platoon

The ego vehicle begins the disengaging procedure while the Platoon is active. The system increases the inter-vehicle time gap to the LSG (legal safe gap) in comparison to the previous one and, once reached, returns to initial state.

1. The ego vehicle switches to "Leave" mode.
2. The start of the ego vehicle's leaving procedure is broadcast to the other Platoon members via V2V communication (message to be inserted).
3. The leaving vehicle starts to increase the inter-vehicle gap in relation to the platoon.
4. When a vehicle departs, the vehicle's Platooning communication is severed.
5. When a vehicle leaves a platoon, it enters "Stand-alone" mode.

C. State Machines – Seda Sensoy

In our project we specified 2 state machines for Platooning concept: Vehicle state (Fig. 4.) and Platoon state (Fig. 3.). In terms of Platoon state, the state changes based on the aforementioned use cases. Initially, the platoon is in “Standalone” state. When the platoon enabler (admin) triggers “Form Platoon” event then the platoon goes into “Platoon Formation” state. Next, when Platoon if formed, “Platoon Formed” event will be triggered and it will enter “Platooning” state where there are three substates: “Steady State Platooning”, “Emergency Breaking”, “Gap Adaptation”. These substates can interchange between each other depending on the situation. However, there is also a special state - Emergency Break after which the platoon is split and because of that it enters “Standalone” state. When “Engage” event is triggered then the platoon goes into “Engaging” state. If the decision of platoon is positive then “Truck Joined” event will be triggered, otherwise “Join Cancelled” event will happen, and both of these events end up with entering “Platooning” state again. During “Platooning” state, when the truck “Truck Leave” event is detected then the platoon enters into “Disengaging” state. After the truck left event is detected in that state, the platoon goes back to “Platooning” state. On the other hand, when it comes to Vehicle state, it starts with “Standalone” state. The next state is “Driving” state, where the vehicle follows steady state driving patterns. From the “Driving” state the vehicle can go into 3 states: “Engaging”, “Leaving”, “Emergency Breaking”. When the “Join” event is executed the vehicle enters into “Engaging” state. When the vehicle joins platoon, “Joined” event is triggered and the vehicle goes back into “Driving” state. On the other hand, when the truck is not allowed to join the platoon, then “Not Joined” event will be triggered and it will return to “Driving” state again. A similar algorithm is valid for “Leaving” state too. When “Leave” event is triggered then the vehicle enters “Leaving” state. After that “Left” event is triggered, the vehicle “Driving” state. In addition, the vehicle can enter “Emergency Breaking” state when the “Emergency Break” event is triggered. However after breaking completed, “Braked” event occurs and the vehicle can enter either of two states “Driving” or “Standalone” based on the vehicle speed. If the speed is greater than 0, then the vehicle enters “Driving” state, otherwise (speed is 0) then the vehicle enters “Standalone” state.

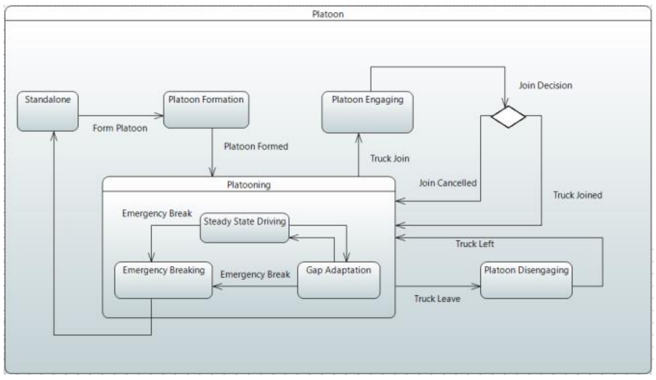


Fig. 3. Platoon State Machine Diagram

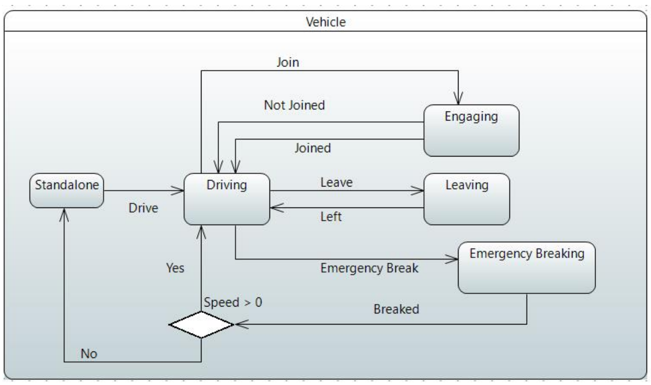
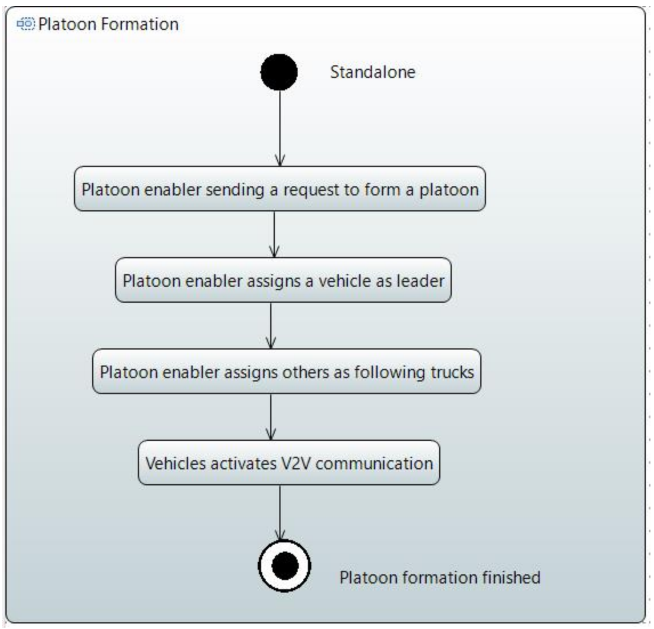


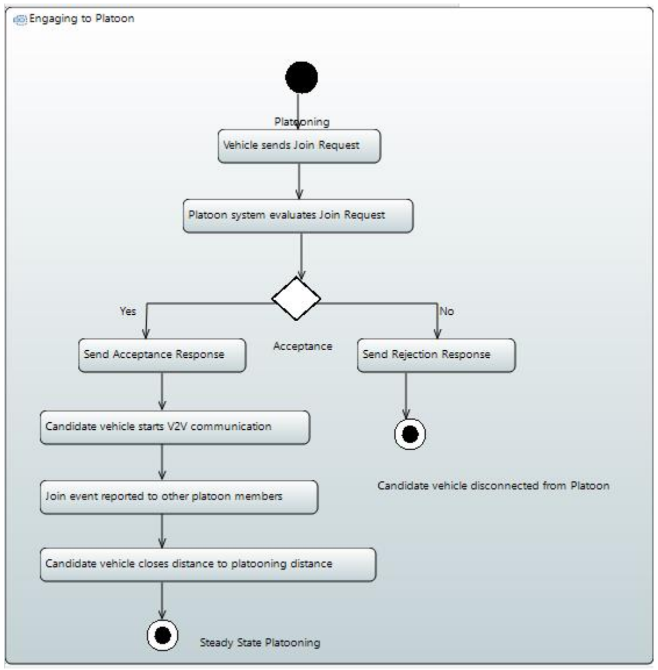
Fig. 4. Vehicle State Machine Diagram

D. *Behavior – Rahim Hashimov*



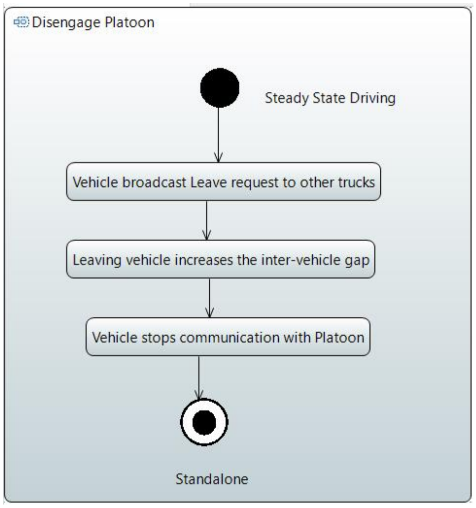
Platoon Formation
Standalone
Platoon enabler sending a request to form a platoon
Platoon enabler assigns a vehicle as leader
Platoon enabler assigns others as following trucks
Vehicles activates V2V communication
Platoon formation finished

Fig. 5. Platoon Formation Activity Diagram



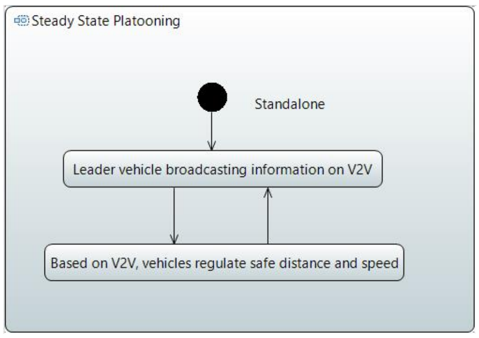
Engaging to Platoon
Platooning
Vehicle sends Join Request
Platoon system evaluates Join Request
Yes
No
Acceptance
Send Acceptance Response
Send Rejection Response
Candidate vehicle starts V2V communication
Candidate vehicle disconnected from Platoon
Join event reported to other platoon members
Candidate vehicle closes distance to platooning distance
Steady State Platooning

Fig. 6. Engaging to Platoon Activity Diagram



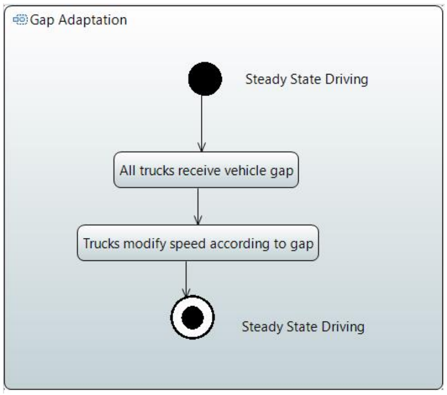
Disengage Platoon
Steady State Driving
Vehicle broadcast Leave request to other trucks
Leaving vehicle increases the inter-vehicle gap
Vehicle stops communication with Platoon
Standalone

Fig. 7. Disengage Platoon Activity Diagram



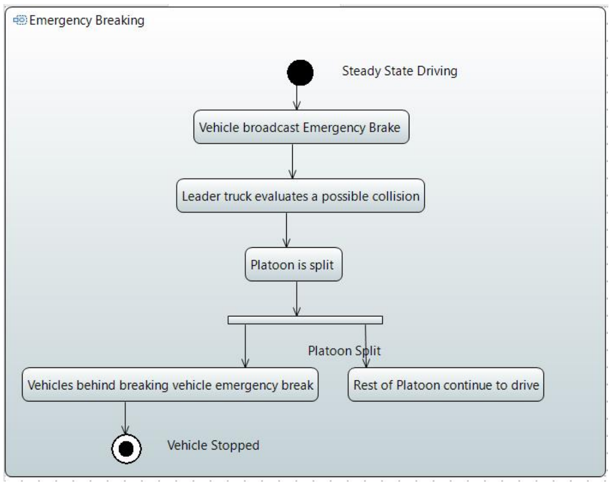
Steady State Platooning
Standalone
Leader vehicle broadcasting information on V2V
Based on V2V, vehicles regulate safe distance and speed

Fig. 8. Steady State Platooning Activity Diagram



Gap Adaptation
Steady State Driving
All trucks receive vehicle gap
Trucks modify speed according to gap
Steady State Driving

Fig. 9. Gap Adaptation Activity Diagram



Emergency Breaking
Steady State Driving
Vehicle broadcast Emergency Brake
Leader truck evaluates a possible collision
Platoon is split
Platoon Split
Vehicles behind breaking vehicle emergency break
Rest of Platoon continue to drive
Vehicle Stopped

Fig. 10. Emergency Breaking Activity Diagram

E. Protocol – Veli Ates

In this part, we will go through the primary communication protocols utilized in truck platoons. The Dedicated Short Range Communications (DSRC) standard was our pick. The DSRC technology has a wide range of applications in the framework of truck platoon communication, notably in truck platoon safety. The Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism is used by the DSRC technology to provide vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communication. It is worth noting that the DSRC apps make use of the Transmission Control Protocol (TCP). Communication failure between vehicles, also known as V2V (vehicle-to-vehicle) communication failure, can occur in truck platooning owing to a number of circumstances such as signal interference, network congestion, or device malfunction. When communication between platooning vehicles fails, the platoon's stability and safety suffer. Control systems like adaptive cruise control (ACC) and automatic braking can be utilized to change the platoon's pace while keeping a safe following distance. These technologies can also be utilized to coordinate the platoon's acceleration and deceleration, decreasing the need for drivers to brake and accelerate manually.

F. Gap Adaptation Algorithm – Veli Ates

The distance to the previous vehicle in truck platooning may be assured by using modern sensor, communication, and control technology. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication technology can also be utilized to increase platoon coordination and response. This enables the platoon to react swiftly to changes in traffic, weather, or road conditions and modify the following distance as needed. Platooning employs a gap adaption algorithm that depends on a mix of sensors, communication systems, and control algorithms to maintain a safe distance from the preceding vehicle. Here is a step-by-step explanation of how this algorithm works:
The lead vehicle or precedence truck transmits its speed and position data to the following trucks in the platoon using a communication system V2X (vehicle-to-everything) technology.

1. Each following vehicle in the platoon employs a variety of sensors, including radar, lidar, and cameras, to identify the location and speed of the trucks ahead of it, as well as any other road obstructions.
2. A control algorithm estimates the optimal distance that the following truck should maintain from the lead vehicle based on data received from the lead truck and sensors on the following truck. This distance is usually measured in seconds.
3. The control system then changes the following vehicle's speed and acceleration in real time to maintain the appropriate distance from the lead truck. If the leading truck accelerates, the trailing truck will likewise accelerate in order to maintain the optimal following distance. If the lead truck slows or stops, the following truck will likewise slow or halt in order to maintain a safe following distance.
4. If the following truck senses an obstruction or danger on the road, such as a stopped vehicle or debris, it will quickly brake and come to a halt, and an alarm will be sent to the lead truck and the other following vehicles in the platoon.

V. Evaluation – Tural Hasanov

A. Implementation

Programming Language
In terms of programming language, we chose Java. Java is a programming language and computer platform that provides a stable foundation for numerous services and applications. Java is still used in the development of new, innovative goods and digital services for the future. Because of its powerful threading paradigm, Java is an excellent language for threading. It has thread synchronization capabilities, making it easier to develop dependable, concurrent applications. Java utilities are particularly designed to make threading easier. Finally, Java provides garbage collection, which simplifies the process of memory management and makes writing thread-safe code easier.

Environment
Eclipse is an integrated development environment. It has a base workspace as well as an extendable plug-in framework for personalizing the environment. It is one of the most widely used IDEs for Java programming.

Data Model
When it comes to the data model for our implementation of the project, we specified classes with relevant attributes (properties and methods), and defined relations between those models. The data model includes the main models: Truck, Platoon and PlatoonServer. Truck and PlatoonServer are executed in threads, that's why they implement "Runnable" interface. Truck model is associated with PlatoonServer via intermediate model Platoon. The relation between these models is:

1. Multiple Trucks can be connected to one Platoon.
2. A Platoon can have one PlatoonServer.
3. Multiple Trucks can be connected to one PlatoonServer over Platoon.

The models have sub models (subclasses) to perform certain operations or operation groups. Truck model has InputHandler, whereas PlatoonServer has ClientPending and EchoClientHandler sub models.
The models (classes) we used in our project:

- **PlatoonServer** – is the server-side controller for a platoon. It handles the incoming requests from the clients (vehicles) and sends responses to them. Its main attributes and functionalities are:
    - Server socket - A server socket listens for network requests. It then conducts some action depending on the request and may deliver a response to the requester.
    - Clients – Vehicles that are connected to the server.
    - Server Port – port number for the server
    - Server Endpoint – server endpoint (URL)
    - Leader Client – the first client connected to the server.
    - Matched Clients – clients that matched by platoon enabler.
    - Clients Pending – clients that are waiting to be accepted by platoon leader.
    - Shutdown – shutdowns the server.

   - Echo Client Handler – client working on separate thread that sends and receives multiple messages from the server.
- Truck
   - ID – Identification number of truck
   - Is Leader – Whether the truck is the leader of the platoon or not.
   - Speed – speed of truck
   - Max Speed – maximum speed that a truck can reach.
   - Acceleration – acceleration of truck
   - Max Acceleration – maximum acceleration that a truck can reach.
   - Max Deceleration – maximum deceleration that a truck can reach.
   - Gap From Platoon
   - Wheel angle – wheel angle of the truck
   - Platoon – platoon instance that truck connected.
   - Platoon Data – data acquired by platoon server.
   - Input Handler – handler for responses coming from server.
   - Client Socket – a client socket to send and receive messages from server socket.
   - Out – message writer
   - In – message reader
   - Emergency Break – emergency break initiation
   - Start connection – connect to platoon server.
   - Send Data Timer – timer to execute sending leader truck data task.
   - Gap Adaptation Timer – timer to execute gap adaptation task.
   - Acceleration Timer – timer to execute acceleration task by given seconds.
   - Accelerate – initiates acceleration task.
   - Acceleration task – handling acceleration of truck
   - Get Coordinates – measuring coordinates of truck with longitude and latitude.
   - Emergency Break – emergency break behavior of truck
   - Start Connection – initiating communication with platoon server.
   - Sending Data Task – handling transmission of truck data to server (acceleration, speed, wheel angle, coordinates)
   - Gap Adaptation Task – adjusting acceleration and speed of truck based on data coming from server.
   - Send Message – send message to server.
   - Stop Connection – stop communication with server.
   - Join Platoon – initiating join event.
   - Leave Platoon – initiating leave event.
   - Send Engage Request – sending join request to server.
   - Send Leaving Request – sending leave request to server.
   - Decide Truck Engage – deciding if new truck accepted to platoon or not.
   - Accept Form Platoon – accepting the platoon formation request coming from server.
- Platoon
   - Server - server instance that platoon is connected.
   - Match Trucks – setting leader truck and matched clients' size.
   - Form Platoon – initiate communication between matched trucks and server.
   - Set leader truck – setting a truck as a leader.
   - Start server – starting platoon server.
- Echo Client Handler
   - Run – receiving requests and transmitting responses to clients.
   - Send message – transmit message.
   - Broadcast – transmit message to multiple clients.
   - Shutdown – close client
- Input Handler
   - Run – receiving responses from server and initiating operations based on those responses
   - Stop – terminate operation of handler.
- Client Pending
   - Client – client that connected to server.
   - ID – ID of truck

Class Diagram

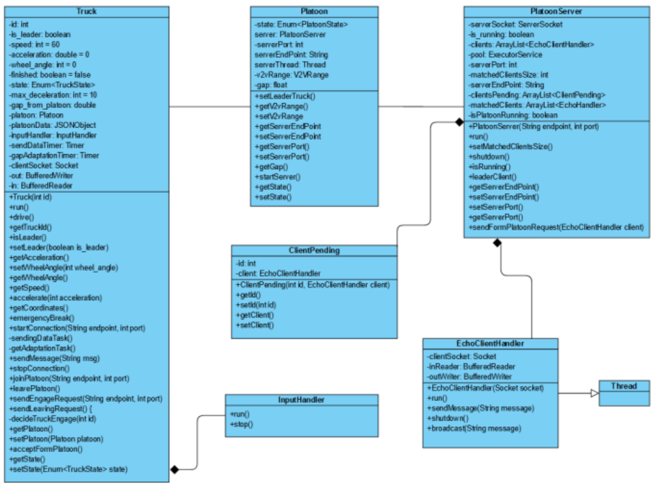


Fig. 11. Class Diagram

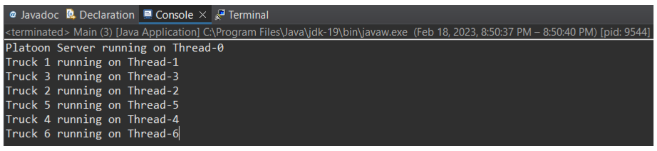
GPU

GPUs, or general-purpose graphics processing units, are specialized processors built for parallel computing workloads. They can be utilized to speed up data processing and decision-making in truck platooning applications. However, GPU programming is often done in low-level languages such as C/C++ and integrating GPU processing with Java-based solutions may be difficult. By leveraging GPU-accelerated libraries, Java bindings for GPU libraries, Java-based GPU programming tools, and GPU-accelerated databases, it is possible to take advantage of the parallel computing power of GPUs in truck platooning implementations.

B. Simulation

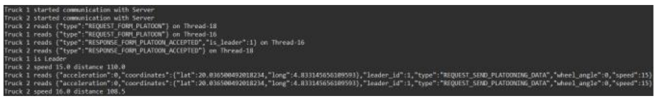
The program consists of sequence of events and states happening after certain seconds:

1. Creating Platoon and Truck instances – when Platoon instance is created a server is also created with given endpoint and port. The server runs on a separate thread and starts listening to client requests. When a truck instance is created, an ID is assigned to that truck and by default the truck is not a leader.
2. Starting threads of trucks
3. Matching trucks and Form Platoon – trucks are added to platoon as matched trucks array. After that the platoon forms platoon. During that process all the trucks inside the matched trucks are iterated and connection between each truck and the server is started. When the client for a truck is created, the server checks the connected client if it exists in matched clients array. The important part here is that if the client is the first client in matched clients then it will be the leader client (leader truck). If yes, then a form platoon request is sent to that client. The truck reads that request via its input handler which runs on separate thread, and it sends acceptance request back to server. The server reads that request and sends form platoon acceptance response back to the truck.
4. Platooning – As soon as getting form platoon acceptance response from server, the leader truck starts sending its data to server continuously in given period. The server accepts that request and broadcasts it to all the clients. Each truck reads that data as JSON object containing acceleration, speed, wheel angle, coordinates. Based on that data each truck performs gap adaptation task on separate threads. The gap is calculated manually by using mathematical equations.
5. Leader truck accelerating for given seconds – truck performs acceleration task. During that task the speed is checked against maximum possible speed of truck. The speed cannot increase higher than maximum speed. On the other hand, the speed cannot go below 0. The task stops operation when the timer of that task is equal to the time input given for acceleration.
6. Leader truck decelerating for given seconds – truck performs acceleration task. The only difference between accelerate and decelerate operations is that the acceleration is negative.
7. Multiple trucks join platoon one after another – each truck sends join request to server. The server reads that requests and forwards it to leader truck. The leader truck reads join request and decides whether to allow the incoming truck or not. Based on the decision it sends acceptance request to server and the server sends acceptance response to incoming truck, as well as other platoon trucks for the sake of information.
8. A truck leaves platoon – the truck sends leave request to server, the server forwards that request to leader truck. Then leader truck sends leave information as a response to other members of the platoon. The leaving truck disconnects from the server.
9. Multiple trucks join platoon one after another – joining operations are performed by new trucks.
10. A truck emergency breaks – a truck sends emergency break request to server. The server forwards that request to the members of platoon which are behind emergency breaking truck.



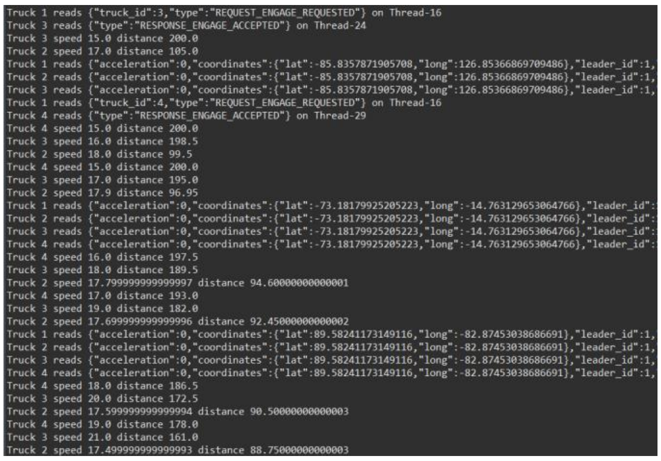
```
<terminated> Main (3) [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe  (Feb 18, 2023, 8:50:37 PM – 8:50:40 PM) [pid: 9544]
Platoon Server running on Thread-0
Truck 1 running on Thread-1
Truck 3 running on Thread-3
Truck 2 running on Thread-2
Truck 5 running on Thread-5
Truck 4 running on Thread-4
Truck 6 running on Thread-6
```

Fig. 12. Starting threads of trucks and server



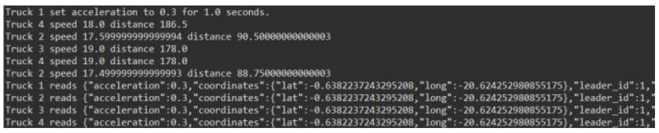
```
Truck 1 started communication with Server
Truck 2 started communication with Server
Truck 2 reads {"type":"REQUEST_FORM_PLATOON"} on Thread-18
Truck 1 reads {"type":"REQUEST_FORM_PLATOON"} on Thread-16
Truck 1 reads {"type":"RESPONSE_FORM_PLATOON_ACCEPTED","is_leader":1} on Thread-16
Truck 2 reads {"type":"RESPONSE_FORM_PLATOON_ACCEPTED"} on Thread-18
Truck 1 is Leader
Truck 2 speed 15.0 distance 110.0
Truck 1 reads {"acceleration":0,"coordinates":{"lat":20.036500492018234,"long":4.833145656109593},"leader_id":1,"type":"REQUEST_SEND_PLATOONING_DATA","wheel_angle":0,"speed":15}
Truck 2 reads {"acceleration":0,"coordinates":{"lat":20.036500492018234,"long":4.833145656109593},"leader_id":1,"type":"REQUEST_SEND_PLATOONING_DATA","wheel_angle":0,"speed":15}
Truck 2 speed 16.0 distance 108.5
```

Fig. 13. Form Platoon



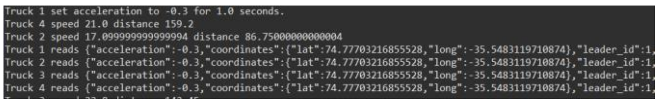
```
Truck 1 reads {"truck_id":3,"type":"REQUEST_ENGAGE_REQUESTED"} on Thread-16
Truck 3 reads {"type":"RESPONSE_ENGAGE_ACCEPTED"} on Thread-24
Truck 3 speed 15.0 distance 200.0
Truck 2 speed 17.0 distance 105.0
Truck 1 reads {"acceleration":0,"coordinates":{"lat":-85.8357871905708,"long":126.85366869709486},"leader_id":1,
Truck 2 reads {"acceleration":0,"coordinates":{"lat":-85.8357871905708,"long":126.85366869709486},"leader_id":1,
Truck 3 reads {"acceleration":0,"coordinates":{"lat":-85.8357871905708,"long":126.85366869709486},"leader_id":1,
Truck 1 reads {"truck_id":4,"type":"REQUEST_ENGAGE_REQUESTED"} on Thread-16
Truck 4 reads {"type":"RESPONSE_ENGAGE_ACCEPTED"} on Thread-29
Truck 4 speed 15.0 distance 200.0
Truck 3 speed 16.0 distance 198.5
Truck 2 speed 18.0 distance 99.5
Truck 4 speed 15.0 distance 200.0
Truck 3 speed 17.0 distance 195.0
Truck 2 speed 17.9 distance 96.95
Truck 1 reads {"acceleration":0,"coordinates":{"lat":-73.18179925205223,"long":-14.763129653064766},"leader_id":
Truck 2 reads {"acceleration":0,"coordinates":{"lat":-73.18179925205223,"long":-14.763129653064766},"leader_id":
Truck 3 reads {"acceleration":0,"coordinates":{"lat":-73.18179925205223,"long":-14.763129653064766},"leader_id":
Truck 4 reads {"acceleration":0,"coordinates":{"lat":-73.18179925205223,"long":-14.763129653064766},"leader_id":
Truck 4 speed 16.0 distance 197.5
Truck 3 speed 18.0 distance 189.5
Truck 2 speed 17.799999999999997 distance 94.60000000000001
Truck 4 speed 17.0 distance 193.0
Truck 3 speed 19.0 distance 182.0
Truck 2 speed 17.699999999999996 distance 92.45000000000002
Truck 1 reads {"acceleration":0,"coordinates":{"lat":89.58241173149116,"long":-82.87453038686691},"leader_id":1,
Truck 2 reads {"acceleration":0,"coordinates":{"lat":89.58241173149116,"long":-82.87453038686691},"leader_id":1,
Truck 3 reads {"acceleration":0,"coordinates":{"lat":89.58241173149116,"long":-82.87453038686691},"leader_id":1,
Truck 4 reads {"acceleration":0,"coordinates":{"lat":89.58241173149116,"long":-82.87453038686691},"leader_id":1,
Truck 4 speed 18.0 distance 186.5
Truck 3 speed 20.0 distance 172.5
Truck 2 speed 17.599999999999994 distance 90.50000000000003
Truck 4 speed 19.0 distance 178.0
Truck 3 speed 21.0 distance 161.0
Truck 2 speed 17.499999999999993 distance 88.75000000000003
```

Fig. 14. Engaging to Platoon 1



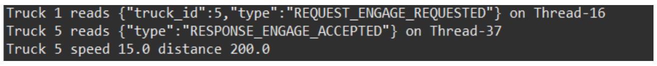
```
Truck 1 set acceleration to 0.3 for 1.0 seconds.
Truck 4 speed 18.0 distance 186.5
Truck 2 speed 17.599999999999994 distance 90.50000000000003
Truck 3 speed 19.0 distance 178.0
Truck 4 speed 19.0 distance 178.0
Truck 2 speed 17.499999999999993 distance 88.75000000000003
Truck 1 reads {"acceleration":0.3,"coordinates":{"lat":-0.6382237243295208,"long":-20.624252980855175},"leader_id":1,
Truck 2 reads {"acceleration":0.3,"coordinates":{"lat":-0.6382237243295208,"long":-20.624252980855175},"leader_id":1,
Truck 3 reads {"acceleration":0.3,"coordinates":{"lat":-0.6382237243295208,"long":-20.624252980855175},"leader_id":1,
Truck 4 reads {"acceleration":0.3,"coordinates":{"lat":-0.6382237243295208,"long":-20.624252980855175},"leader_id":1,
```

Fig. 15. Accelerating



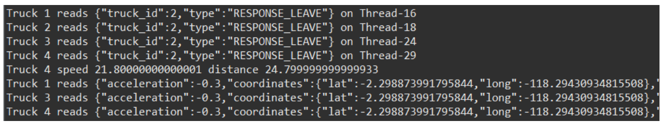
```
Truck 1 set acceleration to -0.3 for 1.0 seconds.
Truck 4 speed 21.0 distance 159.2
Truck 2 speed 17.099999999999994 distance 86.75000000000004
Truck 1 reads {"acceleration":-0.3,"coordinates":{"lat":74.77703216855528,"long":-35.5483119710874},"leader_id":1,
Truck 2 reads {"acceleration":-0.3,"coordinates":{"lat":74.77703216855528,"long":-35.5483119710874},"leader_id":1,
Truck 3 reads {"acceleration":-0.3,"coordinates":{"lat":74.77703216855528,"long":-35.5483119710874},"leader_id":1,
Truck 4 reads {"acceleration":-0.3,"coordinates":{"lat":74.77703216855528,"long":-35.5483119710874},"leader_id":1,
```

Fig. 16. Decelerating



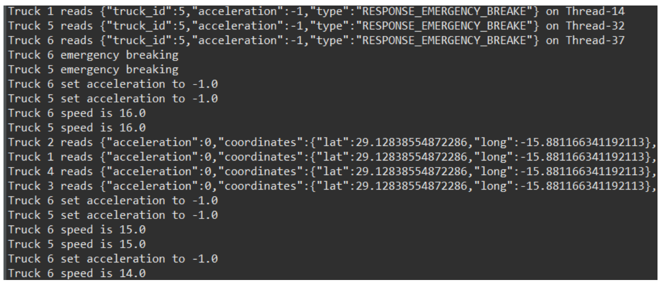
```
Truck 1 reads {"truck_id":5,"type":"REQUEST_ENGAGE_REQUESTED"} on Thread-16
Truck 5 reads {"type":"RESPONSE_ENGAGE_ACCEPTED"} on Thread-37
Truck 5 speed 15.0 distance 200.0
```

Fig. 17. Engaging to Platoon 2



```
Truck 1 reads {"truck_id":2,"type":"RESPONSE_LEAVE"} on Thread-16
Truck 2 reads {"truck_id":2,"type":"RESPONSE_LEAVE"} on Thread-18
Truck 3 reads {"truck_id":2,"type":"RESPONSE_LEAVE"} on Thread-24
Truck 4 reads {"truck_id":2,"type":"RESPONSE_LEAVE"} on Thread-29
Truck 4 speed 21.800000000000001 distance 24.799999999999933
Truck 1 reads {"acceleration":-0.3,"coordinates":{"lat":-2.298873991795844,"long":-118.29430934815508},
Truck 3 reads {"acceleration":-0.3,"coordinates":{"lat":-2.298873991795844,"long":-118.29430934815508},
Truck 4 reads {"acceleration":-0.3,"coordinates":{"lat":-2.298873991795844,"long":-118.29430934815508},
```

Fig. 18. Leaving the Platoon



```
Truck 1 reads {"truck_id":5,"acceleration":-1,"type":"RESPONSE_EMERGENCY_BREAKE"} on Thread-14
Truck 5 reads {"truck_id":5,"acceleration":-1,"type":"RESPONSE_EMERGENCY_BREAKE"} on Thread-32
Truck 6 reads {"truck_id":5,"acceleration":-1,"type":"RESPONSE_EMERGENCY_BREAKE"} on Thread-37
Truck 6 emergency breaking
Truck 5 emergency breaking
Truck 6 set acceleration to -1.0
Truck 5 set acceleration to -1.0
Truck 6 speed is 16.0
Truck 5 speed is 16.0
Truck 2 reads {"acceleration":0,"coordinates":{"lat":29.12838554872286,"long":-15.881166341192113},
Truck 1 reads {"acceleration":0,"coordinates":{"lat":29.12838554872286,"long":-15.881166341192113},
Truck 4 reads {"acceleration":0,"coordinates":{"lat":29.12838554872286,"long":-15.881166341192113},
Truck 3 reads {"acceleration":0,"coordinates":{"lat":29.12838554872286,"long":-15.881166341192113},
Truck 6 set acceleration to -1.0
Truck 5 set acceleration to -1.0
Truck 6 speed is 15.0
Truck 5 speed is 15.0
Truck 6 set acceleration to -1.0
Truck 6 speed is 14.0
```

Fig. 19. Emergency Breaking

VI. Summary – Seda Sensoy

This report aims to provide a detailed summary and conclusion about the benefits, challenges, and potential implications of truck platooning. The primary advantages of truck platooning are higher fuel efficiency, lower emissions, increased safety, and improved traffic flow. The main objectives of the project include developing a reliable and efficient communication system, implementing an algorithm for platoon formation and management, and optimizing the speed and distance between the trucks. The project's communication system employs a message-passing interface (MPI) to permit real-time data and command transmission between the vehicles. The platoon formation and management algorithm is based on a leader-follower strategy, in which the lead truck determines the platoon's speed and direction, while the following trucks maintain a safe distance and change their speed accordingly. The speed and distance optimization technique use a genetic algorithm to discover the ideal speed and distance combination for each vehicle in the platoon. The platoon levels within the platooning project, as well as the use cases that serve as the foundation for technical details. The major goal is to begin early development of demo trucks. This sample version will need to be modified further following more studies, functional safety testing, and practical experience. The high-level platoon use cases are outlined. Initially, the technique for generating use cases is outlined, followed by the high-level use cases for platoon and extra one-lines / assumptions that may be utilized combined with the specific use cases that may be included of the technical deliverables. The following are the key high-level platoon use cases:

- Platoon Formation
- Engaging to Platoon (join from behind by existing platoon)
- Platooning
    - Steady State Platooning
    - Emergency Breaking
    - Gap Adaptation
- Disengage Platoon (leave)

By way of conclusion, the truck platooning project, which was done in Java using parallel programming, highlights the potential of modern communication and control technologies to enhance the fuel economy and travel time of vehicles traveling in a convoy. The study sheds light on the problems and potential connected with truck platooning, notably in terms of developing reliable communication systems, platoon formation and management algorithms, and speed and distance optimization algorithms. The project's findings indicate that truck platooning might be a viable approach for lowering fuel use and enhancing traffic flow.

VII. Appendix – Rahim Hashimov

A. *Definitions*

1. **Array** – A type of data structure that consists of a collection of items, each of which is identifiable by at least one array index or key.
2. **CPU** - A central processing unit (CPU), often known as a central processor, main processor, or simply processor, is the electrical circuitry that processes computer program instructions.
3. **GPU** - A graphics processing unit (GPU) is a specialized electrical circuit that manipulates and alters memory to speed up the production of pictures to provide display.
4. **Eclipse** - The Eclipse IDE is famous for our Java Integrated Development Environment (IDE)
5. **Ego Vehicle** - The vehicle on which operations are performed.
6. **Emergency brake** – Brake action with maximum possible deceleration
7. **Enum** – A type of data that allows a variable to be a collection of specified constants.
8. **Event** - An event denotes the time instant at which a state transition happens, so that the system is in a different mode before and after the event.
9. **JSON** - An open standard file format and data exchange format that stores and transmits data objects made up of attribute-value pairs and arrays using human-readable language.
10. **Leader truck** - The first and controller truck of a truck platoon.
11. **Platoon** - A group of two or more automated cooperative vehicles in a straight line, maintaining a close distance, typically to reduce fuel consumption due to air drag, to boost traffic safety, and to increase traffic throughput because vehicles are driving closer together and take up less space on the highway.
12. **Platoon candidate** - A vehicle that plans to engage the platoon from the back.
13. **Request** – A client sends a request to a named host that is located on a server. The request's goal is to get access to a server resource.
14. **Response –** A response is sent by the server as an answer to a client.
15. **Socket** - A socket is one end of a two-way communication channel that connects two network-running applications. A socket is associated with a port number in order for the TCP layer can detect the application to which data is being transmitted.
16. **Steady State** - In systems theory, a system or process is said to be in a steady state if the variables that describe its behavior (called state variables) do not change over time.
17. **Thread** - A thread is the shortest series of programmed instructions that a scheduler can handle separately. Multiple threads of a particular process can run simultaneously (through multithreading capabilities).
18. **Use case** - A use-case describes how a system should respond to interactions from the system's user or surroundings, such as other traffic participants or road conditions, under various scenarios.

B. *Acronyms and abbreviations*

1. ACC – Adaptive Cruise Control
2. DSRC - Dedicated Short-Range Communications
3. V2I – Vehicle to Infrastructure
4. V2V – Vehicle to Vehicle
5. V2X - Vehicle to any (where x equals either vehicle or infrastructure)

6. WIFI – Wireless Fidelity

VIII. Annex

A. GitHub Overview

Repository Link - https://github.com/TuralHasanov117/--DPS-Semester-Project-Agartha--

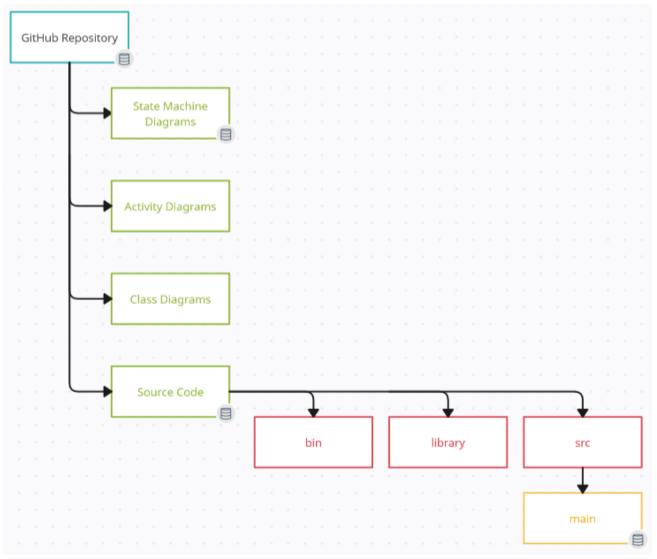
Lines of code:

- Main.java – 120
- Platoon.java – 127
- PlatoonServer.java – 285
- Truck.java – 451

Commits:

- Tural Hasanov – 4
- Veli Ates – 2
- Seda Sensoy – 1
- Rahim Hashimov – 2

Structure:



IX. Affidavit

We (Tural Hasanov, Seda Sensoy, Rahim Hashimov, Veli Ates) herewith declare that we have composed the present paper and work ourself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to then statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form has not been submitted to any examination body and has not been published. This paper was not yet, even in part, used in another examination or as a course performance.

Tural Hasanov: [illegible]

Seda Sensoy: [illegible]

Rahim Hashimov: [illegible]

Veli Ates: [illegible]