Software Requirements Specification (SRS)

Project CACC++ 2

Authors: Erich Hairston, Dean Dawson, Gabrie Italia, Evan Shoup, Kaela Burger

Customer: Mr. Bill Milam

Instructor: Dr. Betty Cheng

1 Introduction

Cooperative Adaptive Cruise Control++ (CACC++) is a system which expands the current capabilities of Adaptive Cruise Control to make the system safer and easier to use. CACC++ enables a vehicle to follow a platoon with the use of sensors to detect nearby vehicles and follow safety procedures, while maintaining a constant speed and distance set by the driver.

To achieve an advanced system, key points such as functionality, requirements, constraints, and dependencies highlight the importance of the necessary details in the CACC++ system. Models are included to provide more detailed interpretations, as well as a prototype that demonstrates several scenarios in which the system behaves. The CACC++ system contains subsystems that include sensors, communication, and control of the vehicle to function without the dependency of an operator.

1.1 Purpose

A functional CACC++ relies on its underlying subsystems and allows the interaction between infrastructure and the vehicle with the appropriate actions to such scenarios. The purpose of the software requirements specification is to help the customer understand the team's interpretation of the product and their approach on how to achieve a system that follows the requirements. Several diagrams are provided that describe the CACC++ system in detail and how it is used daily to provide a better understanding for the reader. A domain model is included, which maps out the main components of the system. A use case diagram is also added, which models common high level behaviors of the system. Sequence diagrams, which describe the interactions between the classes in the domain model in different scenarios are also included. And lastly, state diagrams are included which describe how components of our system move through different states and respond to outside messages.

1.2 Scope

CACC++ is a system available in a vehicle that allows the driver to let the vehicle operate independently and follow traffic regulations. To achieve this goal, the CACC++

communicates with nearby vehicles, using sensor detection, and control of the actuators in the vehicle. This expands on the current capabilities of Adaptive Cruise Control with the intent to make it safer. CACC++ is an embedded system for automotive systems that provides convenience to the operator and provides opportunities for optimization as vehicles send each other signals to communicate. When CACC++ is active, the vehicle will be a part of a platoon, follow the lead vehicle and act appropriately by altering the vehicle's distance and speed based on the communication with the lead vehicle. Another case is that under CACC++ the vehicle is the lead vehicle, in which case it will lead the platoon and communicate with the vehicles behind it. A vehicle will be able to join the platoon through input within the cars user interface, and sometimes the user will be prompted on possible opportunities to join a platoon. The system is intended to support acceleration/deceleration, lateral direction, security precautions, and communication within a platoon.

1.3 Definitions, acronyms, and abbreviations

- **VC**: Vehicle Controller, a system within the vehicle that communicates to subsystems and sends information that updates
- **CACC++**: Cooperative Adaptive Cruise Control
- **OEM**: Original Equipment Manufacturer
- **ITS**: Intelligent Transport Systems, an application that aims to optimize traffic efficiency and provide safer traffic management
- **Target**: a vehicle or object in front of the vehicle that is being tracked
- **GUI**: Graphical user interface, the screen and buttons with which the user can interact with

1.4 Organization

The remainder of the paper is organized as follows. Section 1 contains an overall introduction to the system and its scope. Section 1 further describes the purpose, scope, definitions and abbreviations, and organization of the entire paper.

Section 2 contains a high level description of the CACC++. The description includes the perspective and function of the product, as well as expected user characteristics. Assumptions made are discussed, as well as constraints on the system. Lastly, requirements which have been approportioned are listed.

Section 3 outlines all of the requirements of the system. The requirements are hierarchically organized and listed in order of importance.

Section 4 contains multiple diagrams and models which describe CACC++. Firstly, a Use Case Diagram is shown and described. Then, a Domain Model follows along with detailed data dictionaries to describe the classes within the diagram. Third is Sequence Diagrams which outline some important scenarios within the system. Lastly, State Diagrams are used to describe how the system changes in response to certain triggers.

Section 5 contains a description of the prototype as well as instructions on how to run it. Some scenarios within the prototype are then discussed in detail.

Section 6 contains the references for the paper, including the link to the team website which contains the prototype.

Finally, Section 7 contains contact information for the Instructor of the course for which this document is being created.

2 Overall Description

The following sections will describe at a high level how the CACC++ System functions. Section 2.1 contains the context of the system, as well as the constraints on our system. Section 2.2 outlines the main functionality of the system. Section 2.3 describes the average user of our system as well as their expected experience level. Section 2.4 further discusses constraints on the system. In Section 2.5, key assumptions about our system environment will be discussed. And lastly, in Section 2.6 approportioned requirements will be described.

2.1 Product Perspective

The CACC++ system is a product to be used in vehicles to expand the current capabilities of Adaptive Cruise Control, with the goal to increase safety. The CACC++ system is made of many smaller subsystems which it coordinates and controls. So, while the CACC++ is an independent system within the vehicle, it still must coordinate with all other major systems within the vehicle.

There are many ways in which a user will interact with the system. A user will be able to use buttons on the steering wheel or utilize a center console screen to turn the system on and off. Furthermore, they can make necessary adjustments to the preferred following speed and distance. In addition, the user will be able to use the brakes to disengage the system. Lastly, through the GUI, the user is able to receive feedback from the system.

As for hardware constraints, the vehicle must have actuators installed on the throttle and the brakes for the purpose of monitoring and controlling the acceleration of the vehicle. In addition, a radar and camera must be installed at the front of the vehicle to receive information about targets. Other hardware constraints include a GPS system and a radio to communicate with other vehicles. Lastly, a vehicle controller and independent monitoring system must be present in the vehicle for the system to work.

Every subsystem must have software to allow for operation and communication within the system. The vehicle controller must also have software to manage all of the subsystems, in addition to software for an independent monitoring system. There must also be software to handle communications from external sources within the radio.

The last major constraint of the system is memory. As the system will be embedded on a vehicle, the size of the vehicle's memory limits the system size and the amount of space available for data. This constraint in memory will subsequently affect the software constraints as well. Lastly, certain safety critical systems require a backup memory to allow for system updates to be more easily carried out during operation.

2.2 Product Functions

The CACC++ system consists of many different subsystems that are utilized at different times depending on what the vehicle needs to do. These include acceleration, braking, and communication with other vehicles. The CACC++ system handles these

subsystems using sensors, radars, and other data to control the vehicle for the user on the roadway. The product also allows the user to set a safe following distance between themselves and other vehicles.

The subsystems help the CACC++ by detecting and communicating with the system so that it will take appropriate action to traffic. The driver can enable the CACC++ system to join or leave a platoon. When the vehicle is part of a platoon, it communicates with other vehicles and uses the information to update the speed and distance to be in uniform with the platoon. The driver can leave the platoon by turning off the system or exiting the lane.

During typical use, the system utilizes the GPS and the radio to send GPS packets to platoon vehicles behind the vehicle. The radio is also used to receive GPS packets from platoon vehicles ahead of it. The information from the GPS is combined with data from the radar and camera to track targets, and determine if they are moving or stationary. The information on the target is then used to determine if the system must accelerate, decelerate, brake, or notify the user of any actions they must take. Other functions of the system include communication with ITS and the OEM cloud, and handling system updates.

2.3 User Characteristics

The user of the system should be a licensed driver that follows all traffic laws. The user must also be able to read and interact with notifications, alerts, and different types of controls on a screen inside the vehicle. The user is also expected to choose and maintain both the safe following distance and traveling speed when using the system. Lastly, the user must be alert and ready to take control of the vehicle in case of an emergency or system malfunction.

2.4 Constraints

As the system controls the acceleration of the vehicle, there are quite a few safety critical properties. Firstly, the system must be independently monitored to ensure that the commands being issued to the different parts of the system are correct. The electronic throttle control and brake by wire must also be fully functional for the system to operate, and if either are determined to be functioning incorrectly, control of the vehicle should be deferred back to the user. If the throttle control and brakes were not operating correctly, unintended acceleration or braking can occur which would put the user at risk.

Another safety critical property is a working radar and camera. In certain situations the camera can be used when the radar fails, but if both the radar and camera are not working properly, then control must be deferred once again to the user. The camera and radar are responsible for determining the speed of the vehicle and classifying the target's speed, location, and whether collision will occur. Any error in any of those responsibilities can put the user in danger and compromises safety.

In addition, if any of the main system components such as the radar, camera, and actuators are determined to be not functioning properly, then the system should not be

allowed to start and take control of the vehicle, as if it does take control, safety will be compromised.

Other constraints include signal timing. Messages from the vehicle controller to the actuators and the UI must be nearly instantaneous. If the vehicle controller must issue emergency braking commands to the brakes and the UI, a significant delay in the receipt of that message can cause a collision. For this reason, the timing of these messages must be nearly instantaneous to ensure safety.

Lastly, parallel operation of the CACC++ with other systems within the vehicle is another constraint of the system. The CACC++ must not interfere with the operation of other systems in the vehicle. In addition, the vehicle controller and independent monitoring system must be able to handle contradicting messages from different systems within the vehicle and determine the correct action to take.

2.5 Assumptions and Dependencies

For the development of the CACC++, the vehicle is assumed to have both acceleration and braking potential, and is assumed to be functional. Additionally, it can be assumed that the previous software for cruise control works and the cooperative portion of the software is installed properly. It is assumed that the system will not operate when the vehicle is on unmarked roads. The user operates the vehicle properly and knows when the system should be in use. In addition, it is assumed that the system only operates when all the subsystems are fully functional, and that any loss in functionality is detected through independent monitoring of our system which will then notify the user and defer control back to them.

2.6 Approportioning of Requirements

In the future, the CACC++ may evolve to include more security measures to keep user and vehicle data safe and allow for more over-the-air updates.

3 Specific Requirements

This section contains a hierarchical list of requirements for the CACC++ system. The requirements are listed in order of importance, and are separated into global invariant, primary, and secondary requirements. The primary requirements are also separated into hardware requirements, software requirements, and security requirements.

1. **Global Invariant Requirements**
   - 1.1. Prevent Injury to passengers, and ensure the system is not at fault for a collision.
   - 1.2. Always attempt to avoid an imminent collision with a target through applying brakes, notifying the user, and ceasing acceleration.
   - 1.3. Allow the user to regain control of the system when needed. The user should never be "locked out" of the system.

1.4. Acceleration should not be applied when the distance to the target is below a safe distance.

1.5. Ensure Safe Operation of the vehicle when the system is active by following speed limits and traffic laws.

1.6. Independent monitoring of the VC commands should always occur to ensure reliability and integrity.

1.7. Diagnose and correct system issues when they occur, to recover to an operational state, which includes returning control to the user.

1.8. The CACC++ must never operate concurrently with the manual operation by the user and must not interfere with the operation of other systems within the vehicle.

1.9. The CACC++ must be able to operate with a platoon, and if the platoon is not present then the system must operate as Adaptive Cruise Control.

2. Primary Requirements

2.1. Hardware

2.1.1. Long Range Radars - To implement adaptive cruise control and monitor possible obstacles in the way of the driver there needs to be a way to “see" a great distance in front of the vehicle.

2.1.2. Short Range Radars - Necessary to detect other vehicles or obstacles in closer proximity to the vehicle. Necessary to have them on the sides and rear of the vehicle to implement lane assist, detect vehicles to the right, left or rear of the vehicle, to detect obstacles such as vehicles, pedestrians or objects when driving in reverse, and to detect when a vehicle is approaching too close to the rear of the vehicle.

2.1.3. Cameras - Needed for rear view assistance when driving in reverse, driver monitoring to ensure a driver is available and ready to take over the vehicle when transitioning between CACC++ and manual control, and even for blind spot visibility. Satellite cameras can also be used to give an aerial overview of the vehicle's position on the road.

2.1.4. Actuator Controls - Used to handle the actual functionality of CACC++ from an operating standpoint. Actuators would be responsible for breaking when approaching a vehicle or obstacle in front of or behind the vehicle, changing the trajectory of the vehicle on curves or to implement lane assist, and to accelerate the vehicle to keep the appropriate speed when on slopes or when needed.

2.1.5. Internal Sensors - Allow the system to know specific statuses of internal parts of the vehicle. There are many sensors already present in cars, but the sensors that are pertinent to CACC++ would be sensors that monitor things such as wheel speed, deceleration, assisted braking, acceleration, and other internal functions.

2.1.6. GPS - Allow the system to receive GPS information on location and speed, as well as information about the roads and routes the vehicle is travelling on. This information will be used in conjunction with other location information to help determine targets and proper speed to maintain on a given road.

2.2. Software

2.2.1. Detect and track targets in front using radar, cameras, and GPS, and determine if they are moving or fixed.

2.2.2. Communicate GPS and packet information from the vehicles in front of you to vehicles behind you using radio.

2.2.3. When following a target vehicle, always attempt to match their speed and maintain a safe following distance.

2.2.4. Ensure that the platoon travels in a coordinated and safe manner, by not travelling at unsafe speeds for the given roads, with safe spacing, and with fully functioning subsystems.

2.2.5. When an inappropriate command is given in the system, flag the issue and send it to the OEM cloud.

2.2.6. When the platoon is decelerating or stopping, allow vehicles with less braking power to begin decelerating first.

2.2.7. When the platoon is accelerating, allow vehicles with less acceleration power to accelerate first.

2.2.8. The VC must be able to collect data from a variety of sensors, controls, radars, etc. and decide what action to proceed with.

2.2.9. There must be a system of fault tolerance that prioritizes specific actions over others. The VC must decide which action is the safest within the context.

2.2.10. The VC needs to have direct control over the actuators that manage the throttle and brakes. Communication with these components must be nearly instant.

2.2.11. The VC must update a GUI in real time to provide visual feedback about the vehicle's current state and surroundings, such as obstacles that need to be avoided, to the user.

2.2.12. The VC must be able to receive input from the user interface (Setting the speed of the vehicle, switching control, etc.).

2.2.13. The VC must effectively manage memory and clean up unused objects that are no longer pertinent to the function of the CACC++.

2.2.14. The VC must be able to identify whether it is in a platoon. If so, it must be able to send and receive envelopes to platoon vehicles behind them. It must also be able to continue to track targets outside of the platoon while it is in the platoon.

2.2.15. The VC must be able to receive software updates while still functioning. If it can update the system while functioning it will, if the system needing updating is safety critical, it should defer control back to the driver and update when it is safe.

2.2.16. When receiving an ITS message, determine if the platoon must stop then take the correct action.

2.3. Security

2.3.1. When receiving a message from ITS or an over the air update, verify that the message is valid and not from a false or malicious source.

2.3.2. The VC must be fully aware and capable of monitoring all the hardware and software components in its domain to ensure that they are functioning properly. A piece of the system that is malfunctioning and altering the perception of the CACC++ needs to be accounted for.

2.3.3. Envelopes sent and received by the vehicle should be encrypted and handled appropriately to ensure that the information they contain has not been altered by an external source.

2.3.4. The system should prevent any external attempt to breach protocol or take control of the CACC++. However, in the case that it is breached, it should have an emergency procedure to either reduce the threat or return control to the user.

3. Secondary Requirements

3.1. Do not overwhelm passenger with too many alerts.

3.2. Maintain ease of use and comfort while also providing a secure environment for the vehicle to function under.

4 Modeling Requirements

This section outlines multiple models which describe the CACC++ system. This includes Use Case Diagrams, Object Oriented Models, Sequence Diagrams, and State Diagrams. Each diagram is described in detail and uses Unified Modeling Language notation.

4.1 Use Case Diagram

The use case diagram below describes all the major use cases in our system. The actors include the driver, platoon vehicle, wireless towers, and cloud, and are shown in the diagram as stick figures. The actors are placed outside of the blue system boundary. Each actor plays a role in the system and has use cases that interact with other actors. The use cases are shown in the diagram as green bubbles, with lines connecting them to their actors. Use cases are connected using dotted arrows. The dotted arrow labeled as ‘extend’ signifies that the use case being pointed to is an extension of the other. For example, Disengaging the System is an extension of Leaving the Platoon. The dotted arrow labeled ‘include’ shows that the use case being pointed to is included in the activity of the other use case. An example is Changing Throttle Power includes Accelerating and Decelerating. Below the diagram is a series of tables which further describes each use case and their relationships.

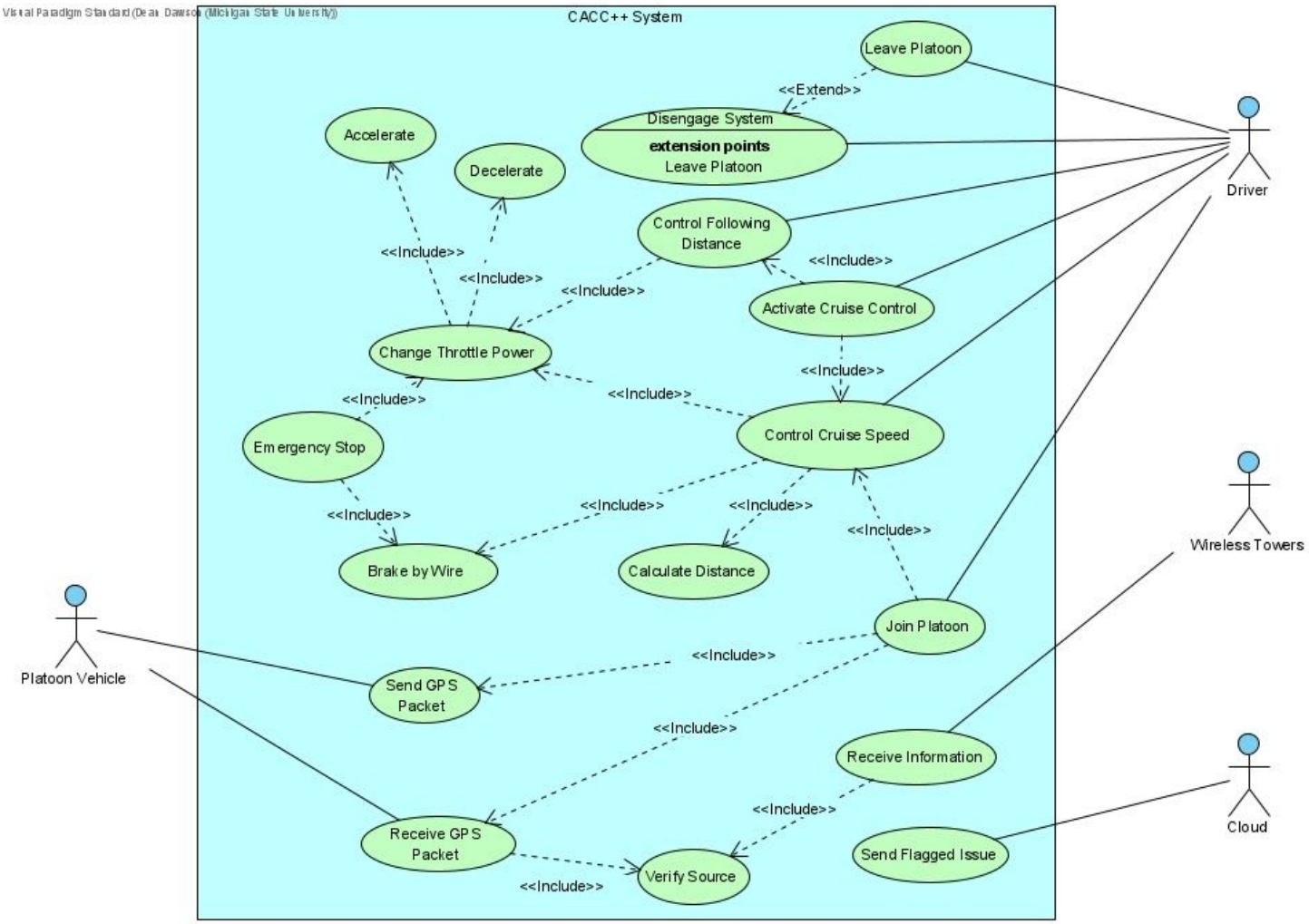

Figure 1: Use Case Diagram

Use Case:	*Activate Cruise Control*
Actors:	*Driver*
Description:	*The driver turns on the adaptive cruise control system and sets a cruise speed and following distance. The system then controls the throttle and brakes to maintain speed and following distance. Slowing down if the target sensed using the radar is too close, and speeding up if the target is too far.*
Type:	*Primary (essential)*
Includes:	*Control Cruise Speed, Control Following Distance*
Extends:	
Cross-refs:	*Requirements 1.5, 1.6, 1.7, 1.8, 2.2.1, 2.2.3, 2.2.11, 2.2.12, 3.2.2.1, 3.2*
Use cases:	*Control Cruise Speed, Control Following Distance*

Use Case:	*Disengage System*
Actors:	*Driver*
Description:	*When the driver signals that the system should turn off, whether by GUI or braking, the system will shut off and defer control back to the driver.*
Type:	*Primary (essential)*
Includes:	
Extends:	*Leave Platoon*
Cross-refs:	*Requirements 1.3, 2.3.4*
Use cases:	*Leave Platoon*

Use Case:	*Join Platoon*
Actors:	*Driver*
Description:	*When a driver chooses or signals that they want to join a platoon, communicate with the other vehicles, and change speed to match the platoon.*
Type:	*Primary (essential)*
Includes:	*Control Cruise Speed, Receive GPS Packet, Send GPS Packet*
Extends:	
Cross-refs:	*Requirements 2.2.2, 2.2.4, 2.2.10, 2.2.14, 3.1, 3.2*
Use cases:	*Control Cruise Speed, Control Following Distance*

Use Case:	*Leave Platoon*
Actors:	*Driver*
Description:	*When the driver chooses, they can leave the platoon. They can either do this by turning off the system or leaving the platoon by exiting the lane. The system may also leave the platoon if a subsystem is failing and operation within the platoon is not possible.*
Type:	*Primary (essential)*
Includes:	*Disengage System*
Extends:	
Cross-refs:	*Requirements 1.3, 1.9, 3.1, 3.2*
Use cases:	*Disengage System*

Use Case:	*Receive Information*
Actors:	*Driver*
Description:	*When passing wireless towers, over the air updates or ITS information may be sent to the vehicle. When receiving this information ensure that it is from a valid source.*
Type:	*Primary (essential)*
Includes:	*Verify Source*
Extends:	
Cross-refs:	*Requirements 1.6, 2.2.13, 2.2.15, 2.2.16, 2.3.1, 2.3.3*
Use cases:	*Verify Source*

Use Case:	*Send Flagged Issue*
Actors:	*Driver*
Description:	*When an issue arises in the vehicle, it should be flagged and sent to the OEM cloud.*
Type:	*Primary (essential)*
Includes:	
Extends:	
Cross-refs:	*Requirement 1.7, 2.2.5, 2.3.3*
Use cases:	

Use Case:	*Receive GPS Packet*
Actors:	*Driver*
Description:	*When in a platoon, GPS Packets are sent to the vehicle from other platoon vehicles over the radio. When receiving packets, we should ensure that the packet is from a valid source.*
Type:	*Primary (essential)*
Includes:	*Verify Source*
Extends:	
Cross-refs:	*Requirements 1.6, 2.2.2, 2.2.13, 2.3.3*
Use cases:	*Verify Source*

Use Case:	*Send GPS Packet*
Actors:	*Driver*
Description:	*When in a platoon, the system should regularly send a GPS packet with location and vehicle information through radio to other platoon vehicles.*
Type:	*Primary (essential)*
Includes:	
Extends:	
Cross-refs:	*Requirements 2.2.2, 2.3.3*
Use cases:	

Use Case:	*Control Cruise Speed*
Actors:	*Driver*
Description:	*The Driver can change the cruise speed through the user interface. The speed can also be changed by joining a platoon or activating the cruise control. This should activate the throttle control if the current following distance allows for it. If the speed is decreased, decrease throttle power. If the speed is increased and the following distance is great enough, increase throttle power.*
Type:	*Primary (essential)*
Includes:	*Change Throttle Power, Calculate Distance, Brake by Wire*
Extends:	
Cross-refs:	*Requirements 1.4, 1.6, 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.2.10, 2.2.12, 3.1, 3.2*
Use cases:	*Change Throttle Power, Calculate Distance, Brake by Wire*

Use Case:	*Control Following Distance*
Actors:	*Driver*
Description:	*The driver can choose to change the following distance when the cruise control is active. The system should use the throttle control to either bring the vehicle closer or further from the target.*
Type:	*Primary (essential)*
Includes:	*Change Throttle Power*
Extends:	
Cross-refs:	*Requirements 1.6, 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.2.10, 2.2.12, 3.1, 3.2*
Use cases:	*Change Throttle Power*

Use Case:	*Verify Source*
Actors:	System (Initiator)
Description:	*When receiving information from outside sources, the system should verify that the source is a valid one.*
Type:	*Secondary, Essential*
Includes:	
Extends:	
Cross-refs:	*Requirements 1.6, 2.2.1, 2.2.3, 2.2.12, 2.2.13*
Use cases:	

Use Case:	*Emergency Stop*
Actors:	*System (Initiator)*
Description:	*If the target is too close and a collision is imminent, the system should begin emergency braking to attempt to stop the collision. This should be done by completely decreasing throttle power and braking by wire.*
Type:	*Secondary, Essential*
Includes:	*Change Throttle Power, Brake by Wire*
Extends:	
Cross-refs:	*Requirements 1.1, 1.2, 1.4, 1.6, 1.7*
Use cases:	*Change Throttle Power, Brake by Wire*

Use Case:	*Brake by Wire*
Actors:	*System (Initiator)*
Description:	*When needed the system can apply braking power to stop or slow down the vehicle.*
Type:	*Secondary, Essential*
Includes:	
Extends:	
Cross-refs:	*Requirements 1.6, 2.2.10*
Use cases:	

Use Case:	*Change Throttle Power*
Actors:	*System (Initiator)*
Description:	*The system needs to change the throttle power to either accelerate or decelerate depending on the situation.*
Type:	*Secondary, Essential*
Includes:	*Change Cruise Speed*
Extends:	
Cross-refs:	*Requirements 1.6, 2.1.5*
Use cases:	*Change Cruise Speed*

Use Case:	*Calculate Distance*
Actors:	*System (Initiator)*
Description:	*When the vehicle is following another vehicle, it's speed should be determined by the speed of the vehicle in front of it so it can maintain a safe following distance to the target.*
Type:	*Secondary, Essential*
Includes:	*Control Cruise Speed*
Extends:	
Cross-refs:	*Reference 1.2, 1.4, 1.6, 2.2.1, 2.2.3, 2.2.8, 2.2.9*
Use cases:	*Control Cruise Speed*

Use Case:	*Accelerate*
Actors:	*System (Initiator)*
Description:	*The system uses the throttle control to increase throttle power, causing the vehicle to accelerate.*
Type:	*Tertiary, Essential*
Includes:	
Extends:	
Cross-refs:	*Requirements 2.1.5, 2.2.10, 2.2.7*
Use cases:	

Use Case:	*Decelerate*
Actors:	*System (Initiator)*
Description:	*The system uses the throttle control to decrease throttle power, causing the vehicle to decelerate.*
Type:	*Tertiary, Essential*
Includes:	
Extends:	
Cross-refs:	*Requirements 2.1.5, 2.2.10, 2.2.6*
Use cases:	

4.3 Domain Model

Figure 2 below shows an object-oriented model, or domain model, for our system. It uses UML class diagram notation for operations and attributes. For example the Vehicle Controller class has many attributes on top such as currSpeed, and many operations below such as setDist(value : int). In addition there are relationships, the line with a diamond on the end signifies aggregation, which means that one class is part of another. For example, the Camera is a part of the CACC++ System. Other relationships are just shown with a line, which can have a multiplicity. Platoon Vehicle Communicates with the Radio, and the Radio can have a relationship with any number of Platoon Vehicles. Every class in the diagram has a data dictionary below which describes in detail the class, its attributes, its operations, and its relationships.

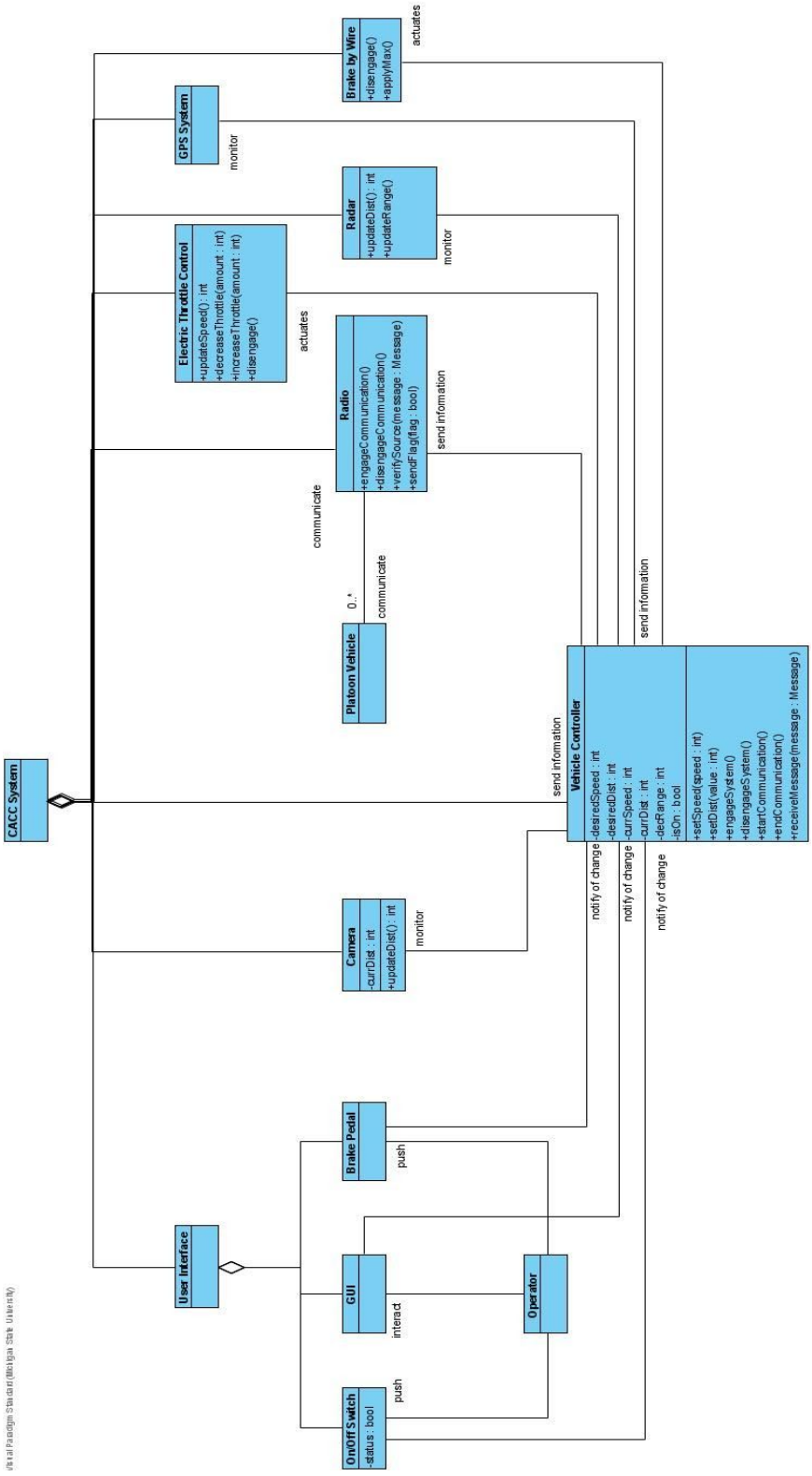

Figure 2: Domain Model for CACC++ System

Element Name		Description
Brake Pedal		Activates the brakes in the vehicle when pressed by the operator. Also disengages the CACC++ System for a vehicle when pressed.
Attributes		
Operations		
Relationships		
	User Interface. Aggregation	
	Operator. The Brake pedal is pushed by the operate to disengage and engage the brakes	
UML Extensions		

Element Name		Description
Brake by Wire		Used by the CACC++ system to electronically apply the brakes for the vehicle.
Attributes		
Operations		
	disengage()	Stops the system from controlling the brakes.
	applyMax()	Applies maximum braking power.
Relationships		
	CACC++ System. Aggregation as Brake by Wire is a subsystem.	
	Vehicle Controller. The vehicle controller uses the brake by wire system to actuate the brakes.	
UML Extensions		

Element Name		Description
CACC++ System		The main system that attempts to maintain constant speed and can be used to set up a platoon of vehicles that follow a lead vehicle.
Attributes		
Operations		
Relationships		
	User Interface. Aggregation. The display on the dashboard the operator can use to change the controls of the vehicle.	
	Camera. Aggregation. The sensor used to estimate a safe distance and speed from the lead vehicle.	
	Vehicle Controller. Aggregation and the system that coordinates with the subsystems.	
	Radio. Aggregation and the radio is used to communicate with platoon vehicles.	
	Electronic Throttle Control. Aggregation. The throttle regulates the speed in which it adds or removes power to maintain the desired speed.	
	Radar. Aggregation and the radar is used as a sensor to detect the objects in front of the vehicle.	
	GPS System. Aggregation. The GPS information is shared between vehicles in the platoon to communicate.	
	Brake by Wire. Aggregation. The brake by wire is electronically applied to the brakes in the vehicle.	
UML Extensions		

Element Name		Description
Camera		Visually identify target vehicle and estimated distance and relative speed.
Attributes		
	currDist: int	Current distance to target vehicle in feet.
Operations		
Relationships		
	Vehicle Controller. The Camera is monitored by the Vehicle Controller.	
	CACC++ System. Aggregation. The camera is a subsystem contained within the CACC++ System.	
UML Extensions		

Element Name		Description
GPS System		Information of a vehicle's location, speed, and direction
Attributes		
Operations		
Relationships		
	Vehicle Controller. The vehicle controller uses the GPS system to communicate location, speed, and direction within a platoon.	
	CACC++ System. The GPS system is a part of the CACC++ system.	
UML Extensions		

Element Name		Description
GUI		Buttons and Screens with which the user can interact with. Input from the user controls different aspects of the vehicle.
Attributes		
Operations		
Relationships		
	Operator. The operator interacts with the GUI to set preferences and control different systems within the vehicle.	
	Vehicle Controller. The GUI notifies the Vehicle Controller of any changes that the operator might make, such as disengaging the CACC++ or modifying the preferred following distance.	
	User Interface. Aggregation. GUI is aggregated from User Interface as it is a part of that subsystem.	
UML Extensions		

Element Name		Description
On/Off Switch		Handles input from the user to activate/deactivate the CACC++ system.
Attributes		
	isOn:boolean	Whether or not the CACC++ system is currently active or not.
Operations		
Relationships		
	User Interface. Aggregate. The interface which the operator interacts with to control the CACC++ system.	
	Operator. The operator interacts with the On/Off Switch, determining the CACC++ system's active state	
	Vehicle Controller. The vehicle controller is notified when isOn is toggled.	
UML Extensions		

Element Name		Description
Operator		The user who is operating the vehicle that the CACC++ system is running on.
Attributes		
Operations		
Relationships		
	On/Off Switch. The interface element which controls the active state of the CACC++ system.	
	GUI. The interface that the operator interacts with to control and monitor the CACC++ system.	
	Brake Pedal. The brake pedal used to decelerate the vehicle.	
UML Extensions		

Element Name		Description
Platoon Vehicle		Another vehicle that communicates with our vehicle via the radio system in the CACC++.
Attributes		
Operations		
Relationships		
	Radio. Platoon vehicles use the radio to communicate with other platoon vehicles.	
UML Extensions		

Element Name		Description
Radar		Determine range, velocity, and angle of objects in front of the vehicle. (e.g., purpose and scope).
Attributes		
Operations		
	updateDist()	Updates the distance between the vehicle and lead vehicle
	updateRange()	Updates detection of the nearby surroundings of the vehicle
Relationships		
	Vehicle Controller. The radar sends position data to the vehicle controller, helping in decision making.	
	CACC++ System. Aggregation. The radar is a part of the CACC++ system.	
UML Extensions		

Element Name		Description
Radio		Convey information to other vehicles and receive incoming information.
Attributes		
Operations		
	engageCommunication()	Begins communication with other platoon vehicles
	disengageCommunication()	Stops communication with platoon vehicles.
	verifySource()	Verifies the source of incoming messages to ensure they are real.
	sendFlag(flag : bool)	Sends a flag to the OEM depending on the value of the parameter given.
Relationships		
	Vehicle Controller. The radio sends/receives data from other vehicles and architecture and passes this information to the vehicle controller.	
	Platoon Vehicle. Other vehicles in the platoon communicate with each other using the radio.	
	CACC++ System. Aggregation. The radio is a part of the CACC++ system.	
UML Extensions		

Element Name		Description
Throttle (Electronic Throttle Control)		Brief description (e.g., purpose and scope).
Attributes		
Operations		
	updateSpeed()	Updates the speed of the vehicle
	decreaseThrottle(amount : int)	Decrease the power exerted on the throttle
	increaseThrottle (amount : int)	Increase the power exerted on the throttle
	disengage()	Disengage from the system
Relationships		
	Vehicle Controller. The vehicle controller uses the throttle to control the speed of the vehicle.	
	CACC++ System. The electronic throttle control is a part of the CACC++ system.	
UML Extensions		

Element Name		**Description**
User Interface		The screens and buttons on the dashboard or steering wheel of the vehicle that can be used to control different aspects of the car. This is the primary way the user will interact with the system.
Attributes		
Operations		
Relationships		
	CACC++ System. Aggregation. The user interface is a part of the CACC++ System.	
	On/Off Switch. Aggregation as the On/Off Switch is a part of the user interface.	
	GUI. Aggregation and the GUI is a part of the user interface.	
	Brake Pedal. Aggregation. The brake pedal is not on the dash but is still part of the User Interface as it is a key component with which the user interacts.	
UML Extensions		

Element Name		Description
Vehicle Controller		Coordinates all subsystems of the CACC++.
Attributes		
	desiredSpeed:int	Desired speed of vehicle.
	desiredDistance:int	Desired distance to target vehicle.
	currDist:int	The current distance to the target.
	currSpeed:int	The current speed of our vehicle.
	decRange:int	The range within which our vehicle should brake.
	isOn:bool	The state of our system, whether or not it is on.
Operations		
	setSpeed(speed : int)	Sets the desired speed of the vehicle.
	setDist(value : int)	Sets the desired distance of the vehicle.
	engageSystem()	Engages the CACC++ system.
	disengageSystem()	Disengages the CACC++ system.
	startCommunication()	Starts communicating with other platoon vehicles.
	endCommunication()	Stop communicating with other platoon vehicles.
	receiveMessage(message : Message)	Receive the message sent through the radio and process it.
Relationships		
	On/Off Switch. The VC is notified of changes by the On/Off switch.	
	GU. The VC is notified of changes by the GUI.	
	Brake Pedal. The VC is notified when the brakes are applied.	
	Camera. The VC monitors the camera so it can figure out how far away the target is.	
	Brake by Wire. The VC actuates the Brakes by Wire to apply different amounts of brake power.	
	Radar. The VC monitors the radar so it can use the data to determine how far away the target is.	
	Radio. The VC sends information using the radio and receives information from the radio.	
	GPS System. The vehicle controller monitors the GPS systems and receives GPS information from it.	
	Electronic Throttle Control. The VC actuates the electronic throttle control to apply different amounts of acceleration power.	
	CACC++ System. Aggregation. The Vehicle controller is a subsystem of the CACC++ System.	
UML Extensions		

4.4 Sequence Diagrams

In this section, many sequence diagrams will be presented to further describe how certain scenarios are handled within our system. The large blue boxes with dotted lines are objects in our system and their life lines. The arrows between the dotted lines represent messages between objects in order which they occur, most recent being the highest on the diagram. Loops are shown by boxes around messages and messages may

have guards, shown in brackets, which are conditions which must be checked before continuing.

Figure 3 below describes a scenario where the vehicle receives a GPS Packet and sends the signal to the vehicle controller. The radio receives a GPS packet from another platoon vehicle radio and then forwards that vehicle controller to process and use the information within the packet.

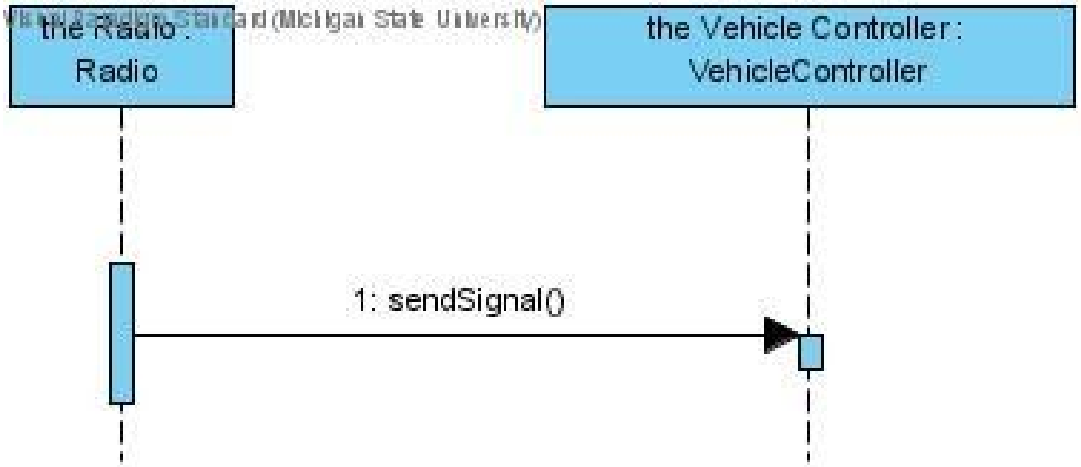

Figure 3: Sequence Diagram for Send/Receive GPS Packet

Figure 4 below describes a scenario where a subsystem potentially fails and the vehicle controller must be able to detect the issue and raise a flag. The vehicle controller and its independent monitoring regularly checks the various subsystems to ensure that they are all functioning properly. If a system is found to be not fully functioning, the vehicle controller sends a flag to the radio so the issue can be sent to the OEM cloud.

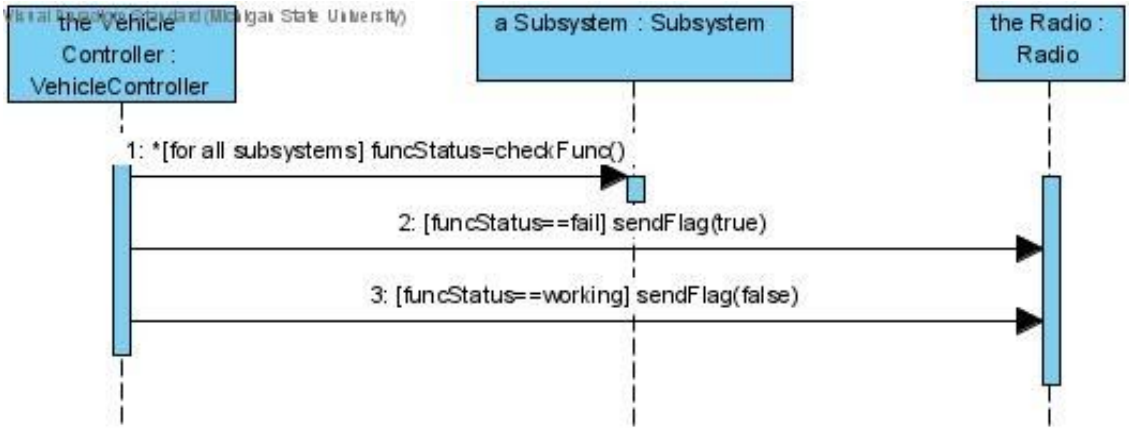

Figure 4: Sequence Diagram for Send Flagged Issue

Figure 5 below describes a scenario where the user activates the cruise control part of the system. The system notifies the vehicle controller that the system needs to be engaged. The vehicle controller monitors the subsystems to control the speed and distance of the vehicle. The speed and distance to the target is managed through changing the acceleration and braking. The proper change is determined based on the value of the current speed and distance to the target of the vehicle in relation to the desired speed and distance.

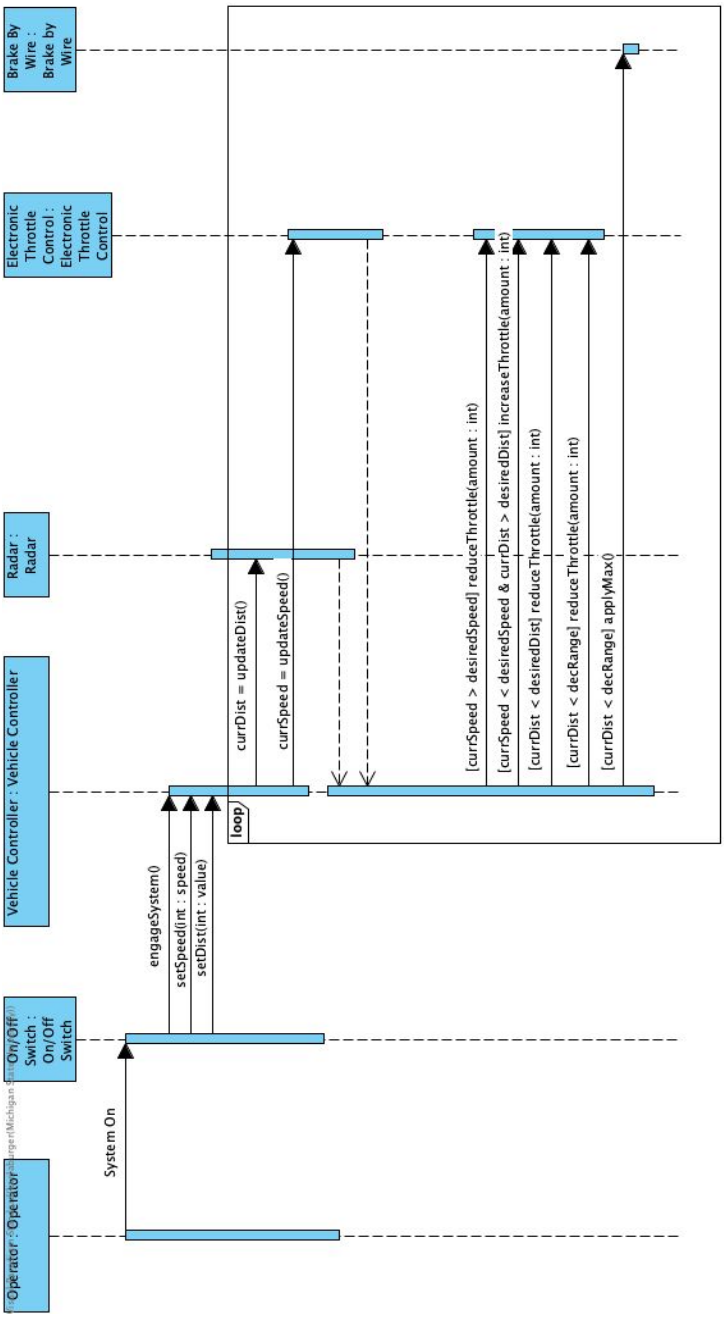

Figure 5: Sequence Diagram for Activate Cruise Control

The sequence diagram below, Figure 6, describes the scenario where the user signals that they want to turn off the system. The user can disengage the system either by hitting the brake pedal or hitting the off switch. After a UI element is pressed, the UI notifies the vehicle controller which disengages all the subsystems, such as the radio and actuators, and defers full control back to the user.

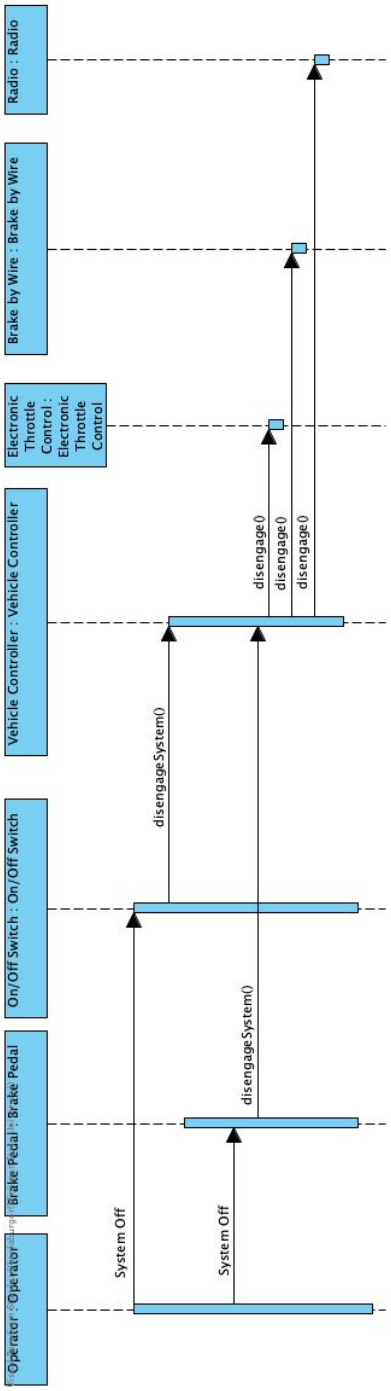

Figure 6: Sequence Diagram for Disengage System

Figure 7 below shows a sequence diagram for joining and leaving a platoon. In both situations, the vehicle controller is notified by the GUI when the user presses the correct button, and takes action depending on whether you are leaving or joining. A user can join a platoon when the system determines platoon travel is possible. The user is given the option to join using the UI in their vehicle. After electing to join, the GUI notifies the vehicle controller. The vehicle controller engages the radio to begin communication with platoon vehicles, and then engages the cruise control if it is not already on. Lastly, the vehicle controller adjusts the cruise speed and the following distance to match the platoon. See Figure 5 for more details on engaging the cruise control part of the system. If the user elects to leave a platoon, the GUI notifies the vehicle controller to end communication, and the vehicle controller notifies the radio to cease communication with the platoon. If the user elects to fully disengage the system, then the vehicle controller disengages other subsystems as well. This was further discussed in Figure 6 above.

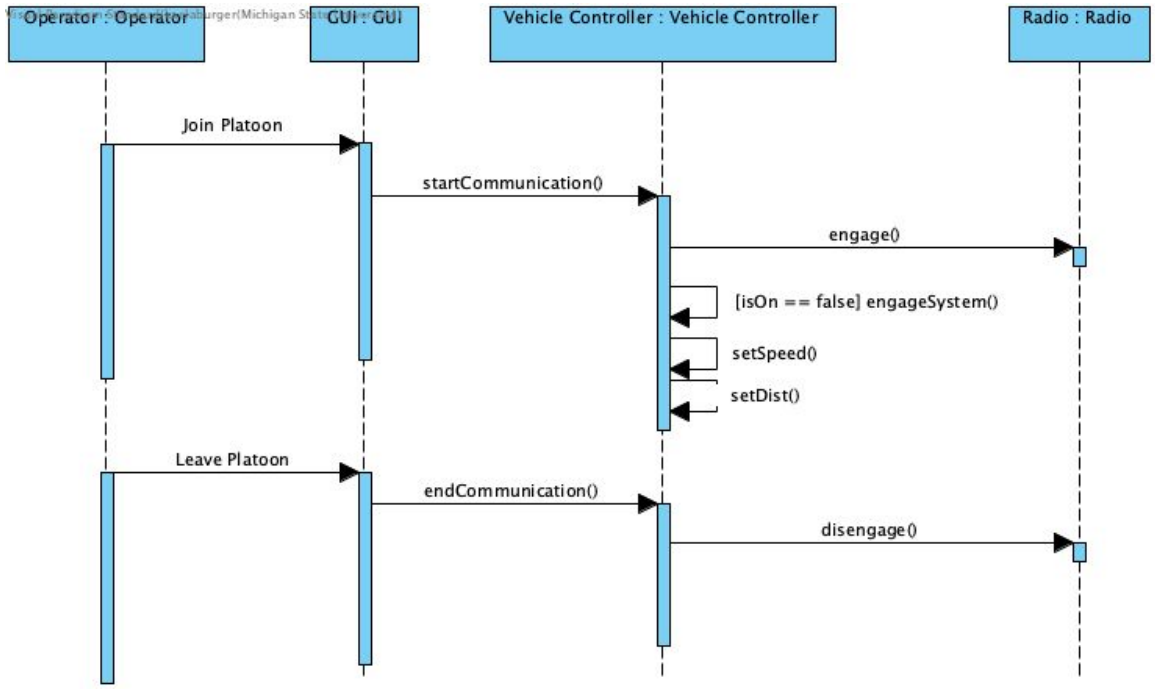

Figure 7: Sequence Diagram for Join Platoon and Leave Platoon

Figure 8 below depicts a scenario where ITS sends an alert to our vehicle. The message is first verified as coming from a valid source by the radio, and then sent to the vehicle controller to handle the message and take the appropriate action.

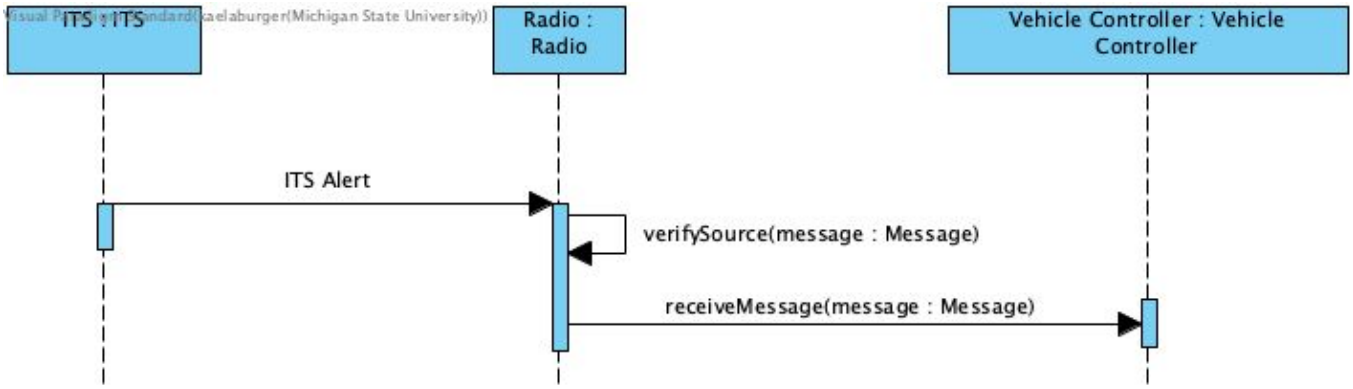

Figure 8: Sequence Diagram for Receiving an Alert from ITS

4.5 State Diagram

This section will contain state diagrams which describe how different components of our system move from state to state. In the following state diagrams, states are represented by the blue rectangles and transitions between states are denoted by arrows. The text on the arrows is the trigger that causes states to change.

In Figure 9, the brake by wire becomes active on system initialization. While the system is active, it is idle and waiting for a signal from the vehicle controller. When a signal is received from the vehicle controller the brake by wire decelerates. When there is no longer a signal from the vehicle controller the brake by wire returns to an active state. When a system failure is detected, an alert is sent to the vehicle controller. The vehicle controller then alerts the system to disengage and move into a restarting state. After restart is complete, the system is re-engaged and moves back to an active state.

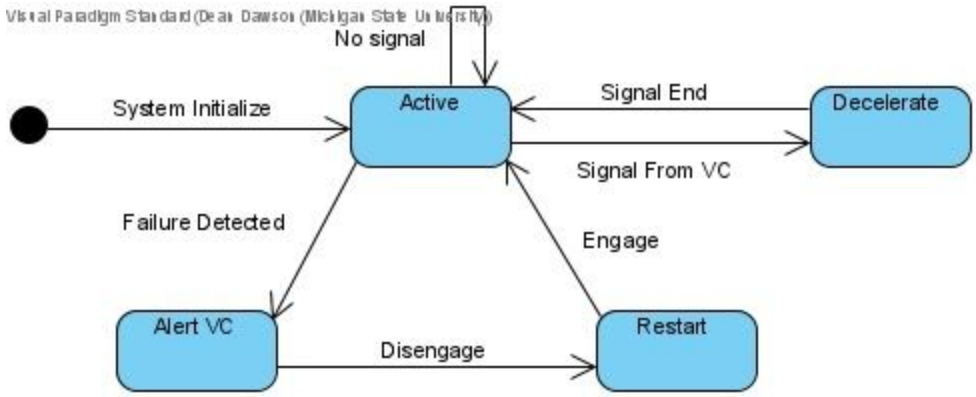

Figure 9: State Diagram for Brake By Wire

In Figure 10, the brake pedal has two states: active and decelerate. When the system is initialized, the brake pedal is active. When the brake pedal is pushed, the brakes will engage and will start to decelerate the vehicle. When the brake pedal is released, the brakes are turned off and the car will continue at its current speed.

Figure 10: State Diagram for the Brake Pedal

In Figure 11, after the system is initialized, the electronic throttle system is active. When the system receives a signal from the VC, it will start accelerating. When this signal ends, the accelerating will stop and will wait for further commands. If a failure is detected, the electronic throttle system will alert the VC, disengage, and restart the system.

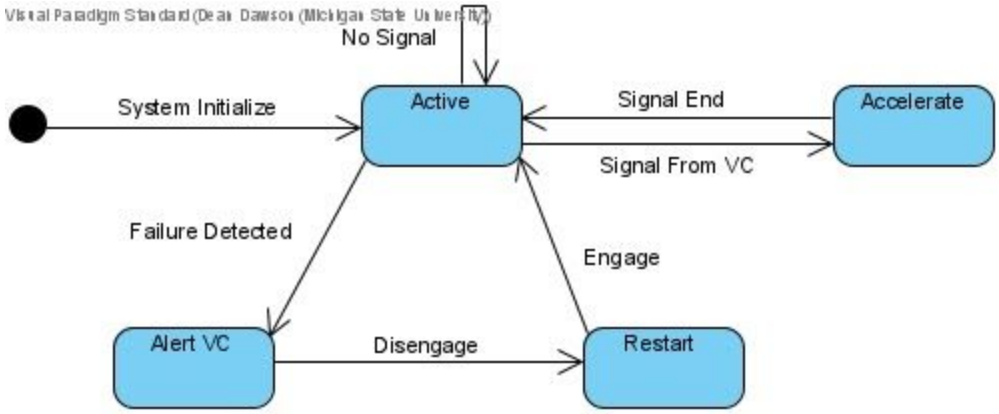

Figure 11: State Diagram for the Electronic Throttle

In Figure 12, the system will be turned on after being initialized. The system will only be turned off when disengaged, and will turn back on after being engaged. If the system detects a failure, it will attempt to restart and engage the system.

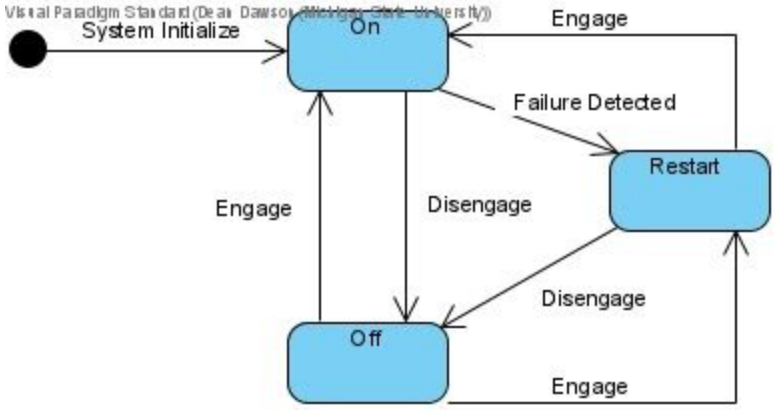

Figure 12: State Diagram for On/Off

In Figure 13, the radar is active after being initialized. If the radar detects an object in the path of the vehicle, it will alert the VC to command other systems to

intervene. If the radar detects a failure, it will alert the VC of this failure, disengage, and attempt to restart the system.

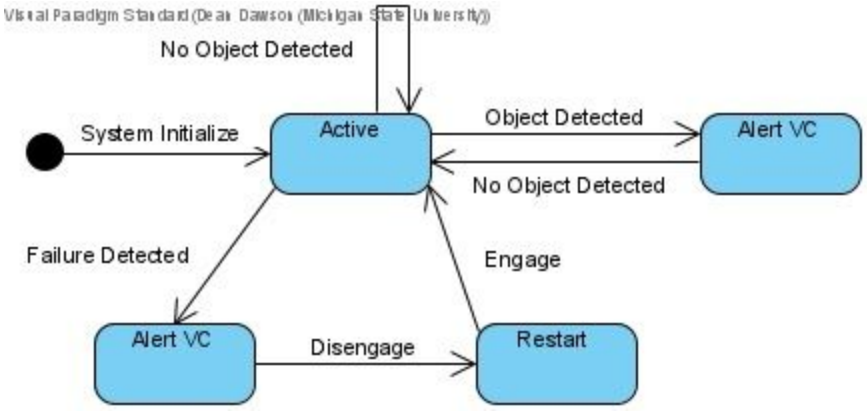

Figure 13: State Diagram for the Radar

In Figure 14, The radio is active after the system has been initialized. When the system is engaged, it will begin communicating with other vehicles utilizing the CACC++ system. Once the radio has received information, it must verify the source of this information and will send a flag once verified. If the radio detects a failure, the system will attempt to restart.

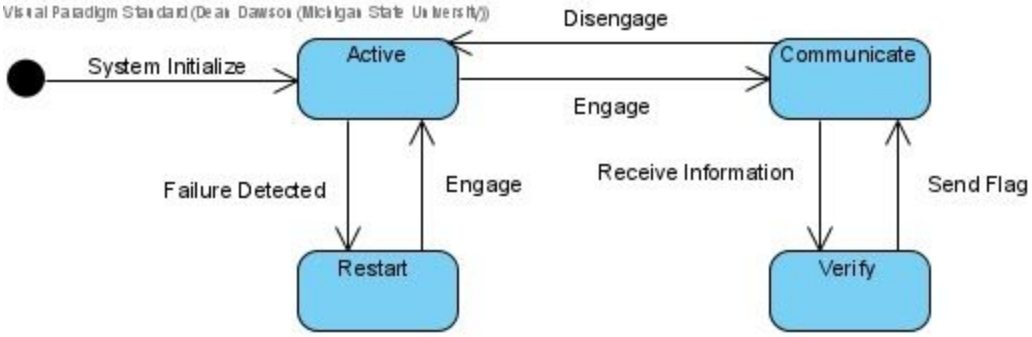

Figure 14: State Diagram for the Radio

In Figure 15, the vehicle controller becomes active on system initialization. The speed and distance can be set while in the active state. When the vehicle controller needs to communicate with another subsystem it starts communication and is in a communicating state and can send messages or return to an active state by ending communication. If a failure is detected by the system the vehicle controller moves to an off state.

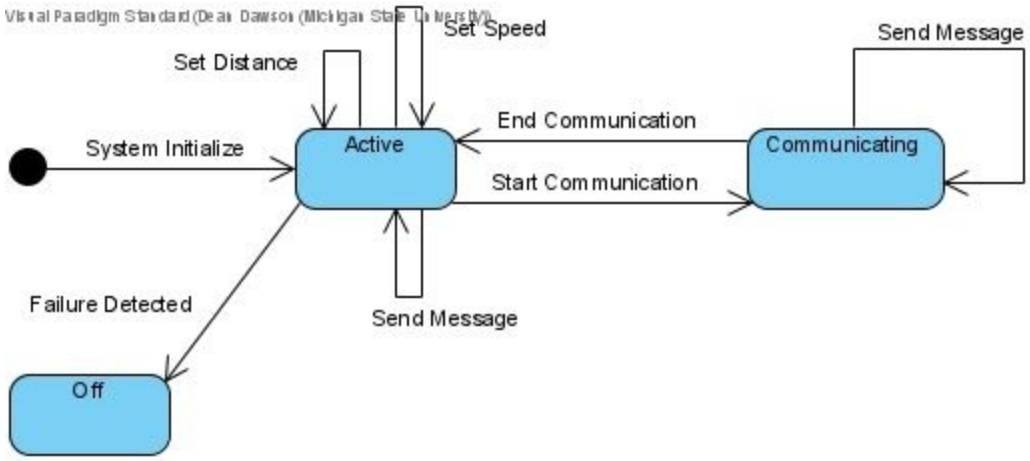

Figure 15: State Diagram for the Vehicle Controller

5 Prototype

In the prototype, multiple scenarios will be presented to exemplify how the CACC++ system will appropriately respond to and handle the given scenarios. There are a total of eight different scenarios that express different concerns that the team should be aware of from a design perspective during the implementation of the system. The prototype explains how these scenarios will be handled and to show that edge cases and possible threats to the functionality of the system as a whole will be accounted for and handled appropriately.

The prototype will show the ability of the VC to adapt to situations and handle any malfunctions or threats to the system, the different scenarios that platoon vehicles may find themselves in, the different components of the system working in cohort to achieve a common goal, and how different terrain conditions can play a role in the functionality of the system.

By viewing the prototype, the viewer should have a strong understanding and good faith in the system's design, integrity, and safety. It should also answer common questions that may be held about the system.

5.1 How to Run Prototype

To run the prototype, navigate to our team website(linked here and in the References Section) and click on the "Prototype WebApp" link. This will run the prototype in your browser. A link to further instructions is available on this site. It should work for all OS. You can also download an executable version of the prototype by clicking on the "Prototype" link which is also located on our team website. Note that the executable prototype will only run on Windows for now.

5.2 Sample Scenarios

The prototype provides a visual display of how the CACC++ system will handle eight different scenarios. To view all eight of the scenarios, navigate to our team website

and open the “Prototype WebApp” link. Here is a single scenario providing an explanation of how the system will handle it for reference:

Scenario 4 - Fully working system as described above. The lead vehicle in the platoon determines an emergency stop is required due to road conditions. Based on the current state of the platoon and the vehicle performance envelopes can the platoon stop or must it divert members to adjacent empty lanes or shoulder?

In Figure 16 below, there is a platoon of three vehicles. They are currently utilizing the CACC++ system to operate as a platoon and communicate with one another to provide helpful data to ensure the safety of the platoon vehicles.

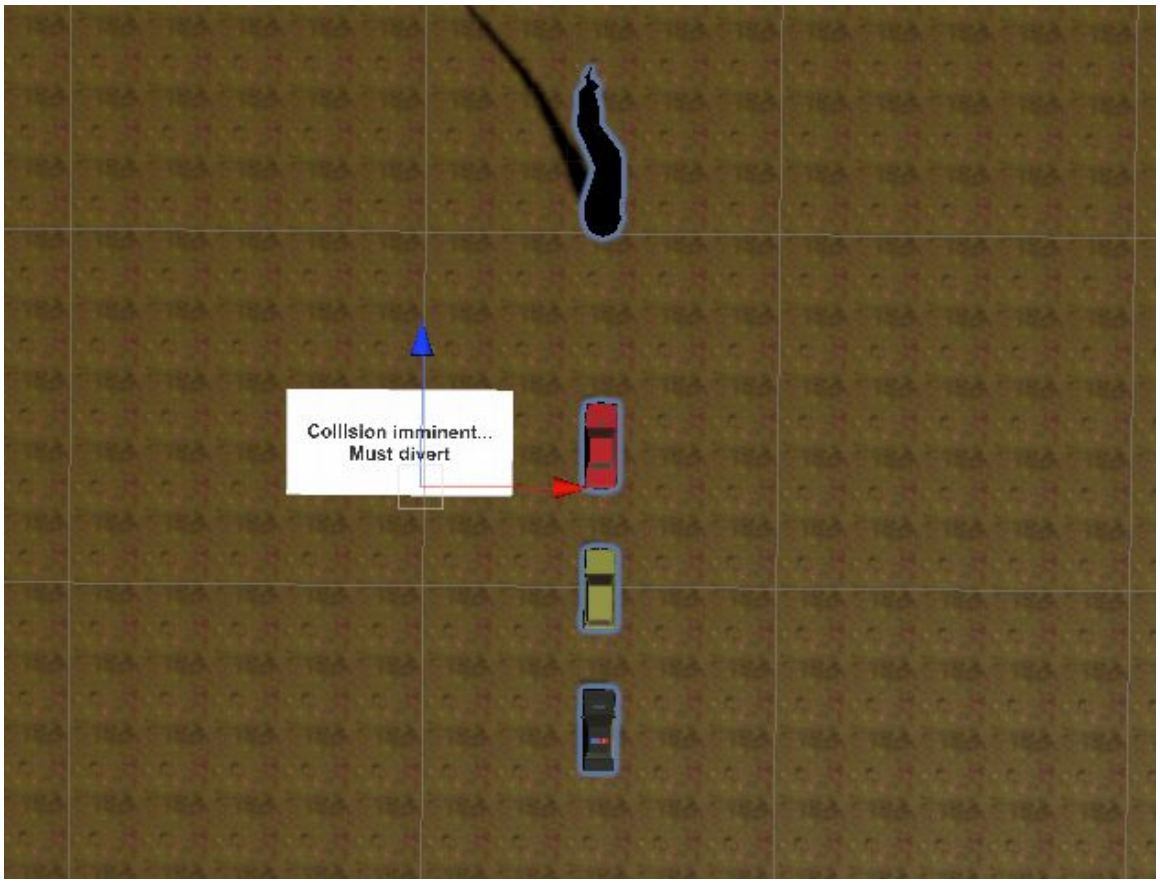

Figure 16: Visual from the prototype.

Observe in Figure 17 below that the platoon is moving forward at a fixed speed with all vehicles a set distance from one another. As the platoon is moving forward, the lead vehicle notices a dangerous obstacle in the way of the platoon. Once the obstacle is identified, the lead vehicle determines the appropriate course of action to take to ensure the safety of the platoon vehicles. The lead vehicle communicates this to the rest of the platoon.

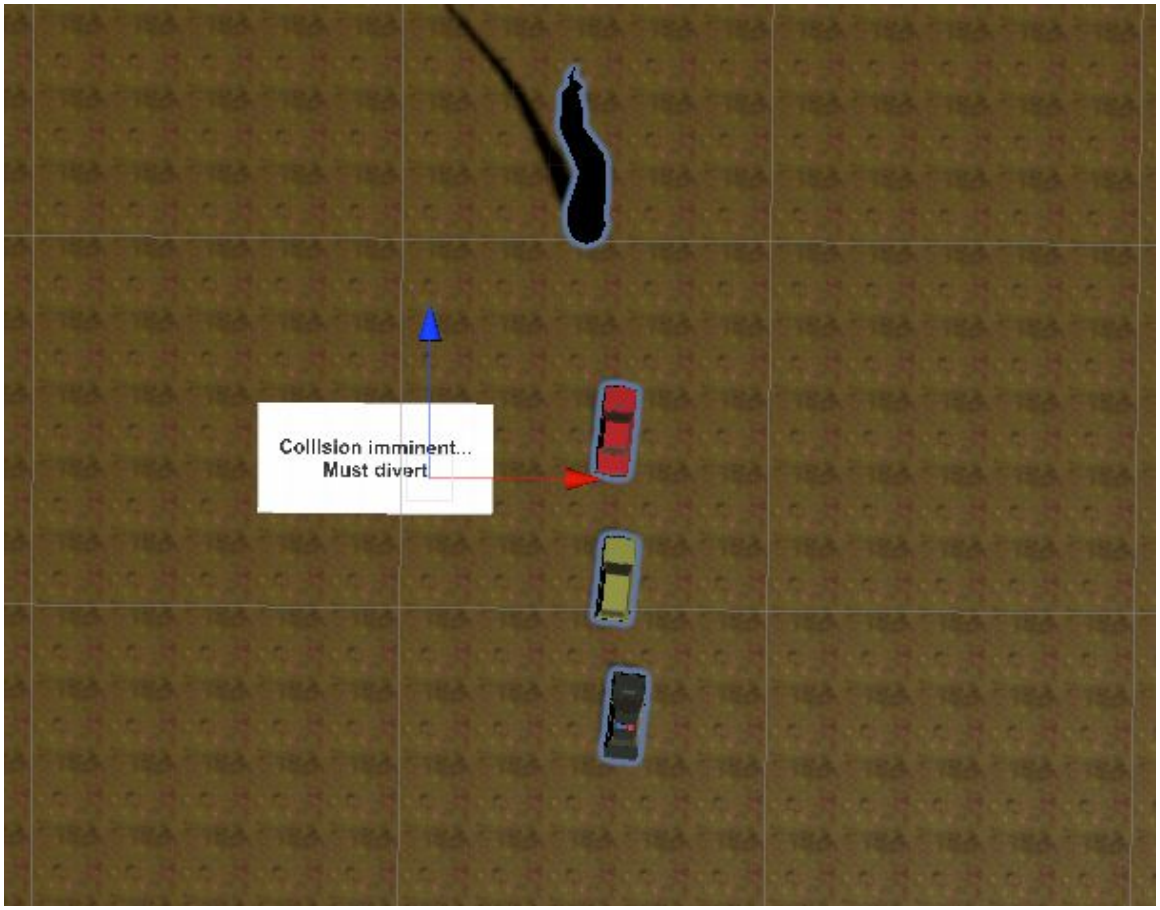

Figure 17: Visual from the prototype.

In this scenario, the correct course of action determined by the driver was to avoid the obstacle by switching lanes. Since the lead vehicle identified the threat early using its radar, the rest of the platoon vehicles were able to know about the threat and divert their course with the lead vehicle. Below in Figure 18, the vehicles are seen diverting to a separate lane or shoulder.

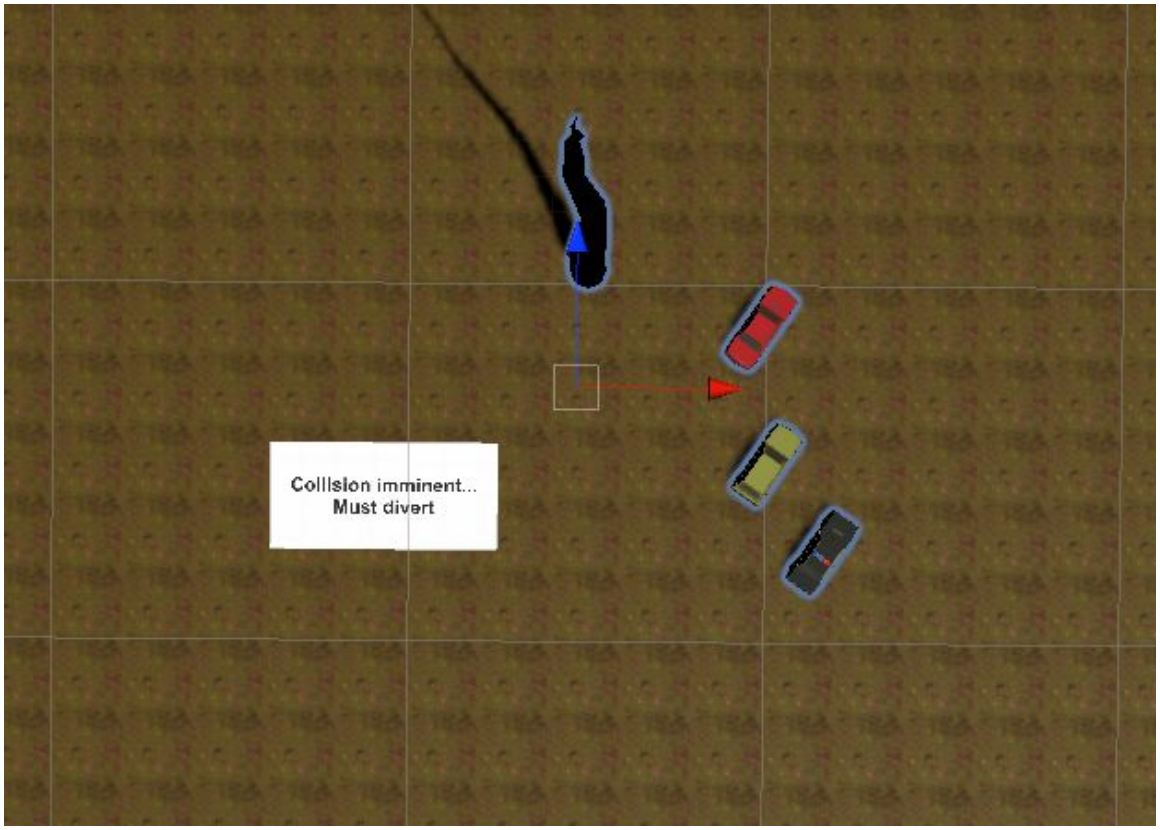

Figure 18: Visual from the prototype.

6 References

[1] A. R. Girard, J. B. de Sousa, J. A. Misener, and J. K. Hedrick. A control architecture for integrated cooperative cruise control and collision warning systems. In Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228), volume 2, pages 1491–1496 vol.2, 2001.

[2] A. Tiganasu, C. Lazar, and C. F. Caruntu. Cyber physical systems - oriented design of cooperative control for vehicle platooning. In 2017 21st International Conference on Control Systems and Computer Science (CSCS), pages 465–470, 2017.

[3] D. Dawson, "CACC++2," *CACC++2 Team Page*, 2020. [Online]. Available: http://cse.msu.edu/~dawson40/. [Accessed: 30-Nov-2020].

[4] F. Bu, H. Tan, and J. Huang. Design and field testing of a cooperative adaptive cruise control system. In Proceedings of the 2010 American Control Conference, pages 4616–4621, 2010.

[5] W. Milam, "Cooperative Adaptive Cruise Control++ (CACC++)," 2020.

7 Point of Contact

For further information regarding this document and project, please contact **Prof. Betty H.C. Cheng** at Michigan State University (chengb at msu.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.