Software Requirements Specification (SRS)

Project: Cooperative Adaptive Cruise Control++

Authors: Duale Mahat, Sasha Morford, Sarah Mostofizadeh, Mitchell Setsma, Katie Sydlik-Badgerow

Customer: Mr. William Milam, Ford Motor Company

Instructor: Dr. James E. Daly

Formatted: Normal

1 Introduction

This Software Requirements Specification document will overview the requirements needed to develop the Cooperative Adaptive Cruise Control++ system. The requirements will include the implementation of several subsystems. The subsystems necessary to implement the system include cruise control, vehicle detection, and driving assist.

This section of this Software Requirements Specification document covers the purpose, scope, and key definitions of the Cooperative Adaptive Cruise Control++ (CACC++) system. The '2 - Description' section of the SRS will specify the perspective, functions, characteristics, and constraints of the product. The 'Description' section will also include the assumptions, dependencies, and requirements which are negligible in the current scope of the product.

The '3 - Specific Requirements' section will outline every requirement which must be implemented to have an outstanding product, and the '4 - Modeling Requirements' section will provide the design and domain for each of these requirements. The '5 - Prototype' section will describe what an ideal prototype would consist of and it will include example designs.

Every outside source that is used in the production of this document will be provided in the '6 - Resource' section. Additionally, the '7 - Point of Contact' section indicates where further information regarding the document can be obtained.

1.1 Purpose

The general purpose of this Software Requirements Definition document is to outline the requirements and design of the Cooperative Adaptive Cruise Control++ system. The intended audience of this document is first the customer, to ensure that the requirements meet their specifications. The document should also be used as a guide to the developers producing the Cooperative Adaptive Cruise Control++ system, to ensure that all requirements have been met.

1.2 Scope

The product to be implemented is the Cooperative Adaptive Cruise Control++ (CACC++) system. This project is a part of the Automated Vehicle Control domain. The application of the CACC++ system's software is to assist the user in maintaining safe control over the motion of the vehicle. The system will assist a vehicle in maintaining a speed that is equivalent to the speed of the vehicle in its front. This automatic speed matching eases the user's strain on long term driving.

The CACC++ should work by maintaining a vehicle speed which is either specified by the user [2] or determined by matching the speed of the vehicle in front of the user's vehicle. The CACC++ system determines and calculates the speed of surrounding vehicles using radar, sensors, and GPS communication [2].

The CACC++ system should decrease the number of rear end collisions, due to safe following distances and speed adjustments. Thus, one benefit of the CACC++ is the increased safety that it provides to both the occupants of the vehicle and people in the vicinity. The system should also benefit the producing company, Ford Motor Company. By creating a system which increases safety, the cost of vehicles with the system implemented can increase. The system should both produce revenue for Ford Motor Company and establish the company as the industry leaders in CACC++.

One objective is developing a CACC++ system that is secure, and thus free from any sort of malicious attacks. Another objective is to implement a platoon system, which is a collection of cars with CACC++ activated and maintains an equivalent speed for each car in the platoon. Thus, the platoon prevents a vehicle in the rear from going faster than the vehicles in front of it and creates an ease of travel for the vehicles in the platoon. The CACC++ system should be reliable and safely react when there are software and/or hardware failures. The system should also be intuitive, easy to use, and aid users while driving in adverse conditions.

The main goals of the project are as follows: produce a system that stays under budget, release adequately tested software, and add value to vehicles through the introduction of the CACC++ system. Value can be added by decreasing the chance of user errors. Another goal is to increase safety for vehicle occupants with this software installed.

The software application will assist the user in maintaining control over the motion of the vehicle in a variety of circumstances. It will do this by making appropriate decisions and acting on them through communication with the hardware. The CACC++ will not function unless the user turns on the system. Not functioning means that the system will not take in sensory data and/or modify the motion of the vehicle. The system's software will only take commands from the user or the system itself.

1.3 Definitions, acronyms, and abbreviations

Table 1: Acronym for the Cooperative Adaptive Cruise Control++ System

Acronym	**Phrase**
CACC++	Cooperative Adaptive Cruise Control++

Table 2: Definitions for the Cooperative Adaptive Cruise Control++ System

Term	**Definition**
Cooperative Adaptive Cruise Control++ System	Defined in the "Cooperative Adaptive Cruise Control++" document as a system which "attempts to maintain a constant forward vehicle speed, as specified by the driver. In addition, using a combination of forward radar and camera sensors, CACC detects when another vehicle (called the target vehicle) is in its forward path and adjusts its own speed via throttle and braking control to maintain a safe following distance behind it." [2]
GPS Information	Information which contains a vehicle's location, speed, and direction. This information is broadcasted to and received by surrounding vehicles. [2]
Platoon	A group of vehicles that follow a lead vehicle. All vehicles receive broadcasted GPS information about the other vehicles and have equivalent speeds. [2]
Radar	A system that detects the presence, direction, distance, and speed of a target vehicle and/or object. [2]
Throttle	A device that controls both the speed and the flow of fuel in an engine. [2]
Safe Spacing	The distance between a target and a trailing vehicle that is considered a safe following distance. [2]
Vision/Camera Sensors	Sensors which capture images that determine the presence of target vehicles and objects in their field of view. [3]

Actuators	Devices that control vehicles' longitudinal accelerations, lateral directions, and vertical displacements. [4]
Supervisory Controller	A controller that determines which intent is most appropriate to be commanded to the actuator. [2]
Target Vehicle	A vehicle that has been detected by the CACC++ system using sensors and radar. This vehicle's speed is detected and matched by a vehicle with the CACC++ system activated. [2]

1.4 Organization

The section of this Software Requirements Specification document introduced the scope and key definitions needed for the Cooperative Adaptive Cruise Control++ (CACC++) system. The '2 - Description' section of the SRS will specify the perspective, functions, characteristics, and constraints of the CACC++ product. The 'Description' section will also designate which tasks are not necessary to be completed in the initial development of the system.

The remainder of the Software Requirements Specification will define the requirements needed to develop the CACC++ and the scenarios that must be accounted for. The '3 - Specific Requirements' section will outline every requirement which must be implemented to produce a successful system. The '4 - Modeling Requirements' section will provide the design and domain for each of the requirements which are listed in section 3. The '5 - Prototype' section will describe an ideal prototype and will example scenarios and responses to the scenarios.

The '6 - Resource' section of the SRS will list every resource that is used in the creation of the SRS. Additionally, the '7 - Point of Contact' section indicates where further information regarding the document and its contents can be obtained.

The Software Requirements Specification document covers the purpose, scope, and key definitions of the Cooperative Adaptive Cruise Control++ system. The SRS will specify the perspective, functions, characteristics, and constraints of the product. It will also include the assumptions, dependencies, and requirements which are negligible in the current scope of the product. The design and domain for each of these requirements will be provided. There will be an overview of an ideal prototype and its design. The SRS also includes a 'Resource' and 'Point of Contact' section.

2 Overall Description

This section gives an overview of the product. How the product interacts with its surroundings will also be discussed in this section. This will include other system dependencies and user interactions.

2.1 Product Perspective

The product is designed for the motor vehicle industry. The CACC++ will be part of the vehicle's larger system. This relationship is shown in Figure 1. The Cooperative Adaptive Cruise Control++ system will be initiated via a button on the vehicle's steering wheel. The interface will not be complex. A simple 'on' and 'off' switch will be provided to the user.

The vehicle and CACC++ will have a software interface. This interface will include speed exchange. The CACC++ will take over speed operations once engaged. The interface will also display warnings outputted from the CACC++ system. The CACC++ will use the new hardware systems: radar and camera sensors. These hardware features will be connected to the vehicle and may be used for safety features in the future. However, currently the camera and radar sensors

will be physically connected and used for location detection for the CACC++ system. The location detections will provide information to determine the correct speed of the vehicle.

Memory will be a constraint that needs to be taken into consideration. In theory, there can be an infinite number of objects that are surrounding the vehicle. Each object will take up memory and have location and dimension characteristics attached to the object. Therefore, there must be a system in place to eliminate the risks associated with memory overflow. Additionally, time is an important constraint for the CACC++, thus the system needs optimized code and proficient hardware. Seconds while driving could be the difference between a user's life or death.

The operation that this system has access to, is speed. Given these constrains, the system must be able to account for all possible scenarios. This includes, but is not limited to, hill terrain, sudden breaking of objects ahead, and vehicles with different braking distances. The users can decide to customize the CACC++ System with their desired minimum car follow distance.

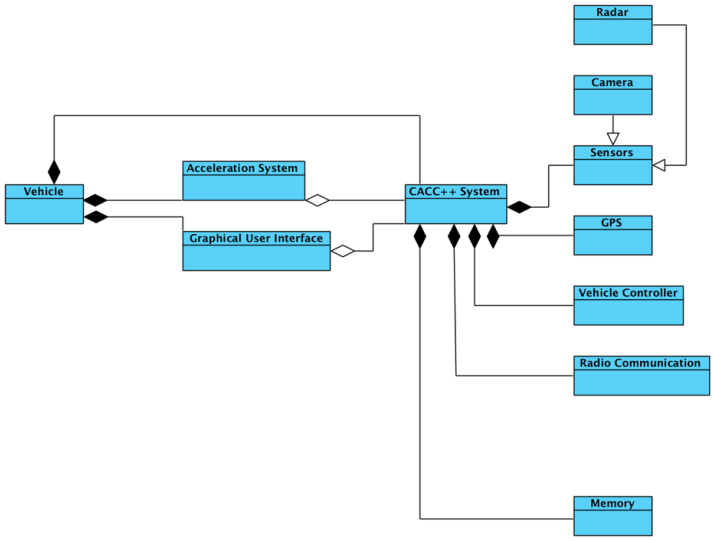

Figure 1: The relationship between the vehicle and the CACC++ system.

2.2 Product Functions

How the product is functions will be determined by simple interactions of systems within the CACC++. The whole purpose of the system is to maintain a constant forward speed, either by matching the speed of a leading vehicle or allowing the user to specify their vehicle's speed. Through a combination of radar and camera sensors, the CACC++ detects a target vehicle in its forward path. This will allow for the adjustment of speed, to maintain a safe following distance. The adjustment is done automatically by the system, through transmitted data, or by user specification.

On the user-display window, the user has the option to match their vehicle's speed to the speed of the vehicle in front of it. The vehicle with a CACC++ system installed also transmits location information to its trailing vehicle, thus creating a platoon of vehicles with safe following distances. The system will utilize encapsulation because the user of each vehicle does not need to know how the GPS information is transmitted, but the user sees all the relevant information on a screen to start the chain of relevant actions.

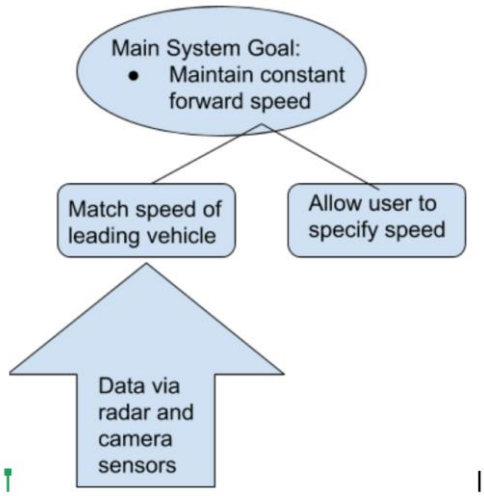

Figure 2: A diagram of high-level goals of the CACC++.

2.3 User Characteristics

The user, a driver, must meet all requirements to be a certified driver in the context of the law that is applicable to the area of operation. The user should be able to understand (through reading or listening) simple information details displayed on a screen such as the location, speed, performance and direction of a target vehicle.

2.4 Constraints

The CACC++ will have the constraints of cost, time, memory, ability to function with current systems, camera technology, radar technology, parallel operations, language differences, and security [5]. The systems in parallel will be the CACC++ and the normal functions of the vehicle.

There will be a cost and time restriction for this feature. As in other car features, this feature will have a value to the customer and thus the company. However, if this the cost exceeds the value, then the feature is not worth implementing. The feature will also have a time constraint. There is a window for the time that this feature will be relevant. If the process is prolonged more risks are added. For example, the risk that the update will become outdated before release.

There are other constraints beyond cost and time that apply to the CACC++ system. The system will be implemented in a functional vehicle. It is important that the new system does not interfere with the current systems. This could risk the safety of the user. For example, the braking system needs to work as it did previously, but with an added feature. The camera and radar technology currently available are also a limiting factor. The system will not have the ability to use radar or camera in a way that has not been developed. The system must be able to run as a parallel operation. Other functions will be running at the same time as the CACC++. It is vital that the CACC++ will not stop the other systems from functioning when it is engaged. The system will also have constraints pertaining to language. The other coding languages used in the vehicle will have to be compatible to the one used for the CACC++. The language used already in the vehicle should be highly considered for use in the CACC++ system. Security will also be a constraint. The CACC++ has access to acceleration. If this system is open to be tampered with, the safety of the user is compromised.

There must be safety features that have been tested and proven effective before the product is released. The CACC++ will automatically control the vehicle, in place of a human driver. There are many unknown actions that a human subconsciously makes when maintaining the direction and speed of a vehicle. When replacing a human operator there is the risk of overlooking one of these actions/procedures. Therefore, it is vital for the system to be extensively tested. Also, this system will be designed to work in adverse conditions. This includes but is not limited to snow, rain, or hail.

The system will not allow the vehicle to operate faster than the posted speed limit. This principle will also apply for speed limit changes in construction zones. This information will be broadcasted from the GPS. The system cannot function on unpaved roads. This is to prevent the system from needing to handle situations where regulated stops occur. This includes situations such as traffic lights and stop signs where the system will not be able to function properly.

2.5 Assumptions and Dependencies

When developing the system, it can be assumed that the car has hardware with the ability to accelerate and break. Additionally, it can be assumed that the previous software for cruise control works. It can be assumed that the system will not be turned on when the vehicle is on unpaved roads. The assumptions about the user interaction are that the user has full driving capabilities and knows when the system should be turned on and off.

2.6 Approportioning of Requirements

For future releases, the use of the cameras and sensors may provide backing up assistance. Additionally, the CACC++ should be able to interact with similar systems that have been produced by other companies. Also, the system will get periodical updates and occasional bug fixes.

3 Specific Requirements

1. **HARDWARE**
    - 1.1. **Sensors**
        - 1.1.1. Radar: The radar detects, identifies, and tracks a target vehicle in its forward path. The radar transmits the speed of the target vehicle.
        - 1.1.2. Camera: The camera identifies an obstacle in the path of the vehicle.
    - 1.2. **Global Positioning System (GPS)**
        - 1.2.1. The GPS collects the vehicle's speed, location, and direction. This is done through communication with a satellite-based data system.
        - 1.2.2. In the case of radar failure, the GPS collects the position of the leading vehicle.
    - 1.3. **Vehicle Controller**
        - 1.3.1. The main vehicle facilitates adjustments to maintain a safe following distance. The controller will communicate with the GPS, the radar, and the camera. The minimum distance between vehicles will be set by the user or automatically calculated by the system.
        - 1.3.2. The vehicle controller also controls all the other subsystems, including but not limited to steering, brake lights, and turn signals.

1.3.3. The controller also triggers commands to the brakes and throttle to ensure a safe following distance is maintained.
1.3.4. The controller implements radio communication between a platoon of vehicles.

1.4. **Radio Communication**

1.4.1. The radio communication sends information between vehicles in the platoon. It also handles the infrastructure, ensuring the broadcast of speed and location of vehicles within the platoon.

2. USER INTERFACE

2.1. **The Graphical User Interface**

2.1.1. The user interface provides the user interaction with the CACC++ system through alerts or warnings directed to the user through text on a screen.
2.1.2. The interface allows visual display of information such as the speed, the target vehicle distance, and the description of target vehicle performance. The performance information includes target vehicle braking and acceleration capabilities in G units of gravity.
2.1.3. The GUI will notify the user in the circumstance where an imminent diversion is needed.

3. SAFETY AND MEMORY

3.1. **Memory**

3.1.1. Memory should not be allocated dynamically, as it will require more time at runtime. Thus, static memory will be used.

3.2. **Security**

3.2.1. In the case of suspected software infiltrations, the system should automatically turn off itself

3.3. **Multitasking control**

3.3.1. In the event where the vehicle controller receives several requests relating to different actions, the controller should be able to decide the requests completion appropriately.

4. USER

4.1. **User/ Vehicle Operator**

4.1.1 The user, provided they met the user characteristics described above, interacts with the GUI. This interaction is completed through an initial button press to start or stop the CACC++ system. The remainder of the interactions are as described in section 5 of the 'Specific Requirements'.

4 Modeling Requirements

The use case diagram of the CACC++ system is as follows:

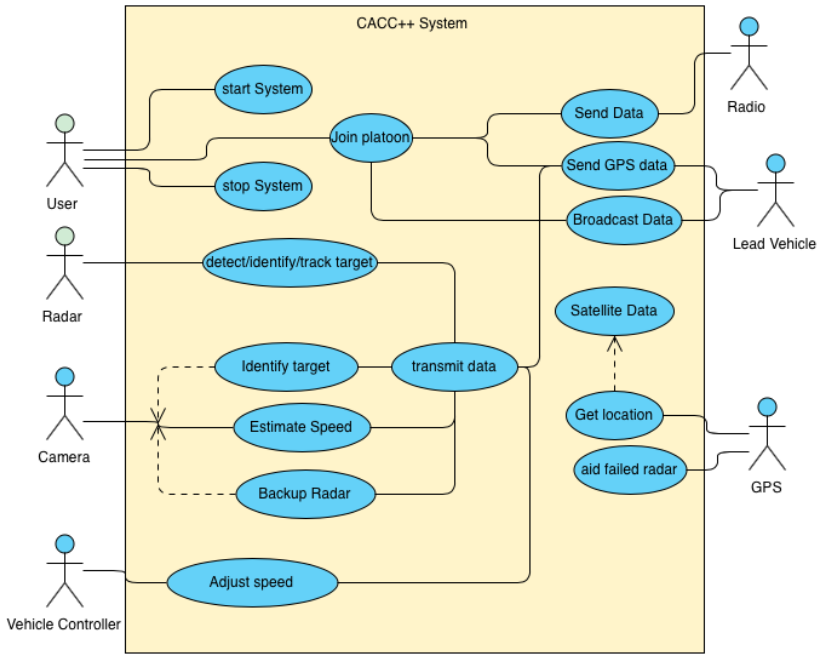

Figure 3: use-case for the CACC++ system

Use Cases

1: Use Case: Start System
Actor: Driver
Type: Primary, essential
Description: The user starts the CACC++ system through a button press "Start" in the Graphical User Interface. This initiates the system and all its related features
Cross References: Requirements: 1,2,3,4

2: Use Case: Stop System
Actor: Driver
Type: Primary, essential
Description: The user stops the CACC++ system through a button press "Stop" in the Graphical User Interface. This stops the system and all its related features.
Cross References: Requirements: 1,2,3,4

3: Use Case: Join platoon
Actor: Driver
Type: Primary, essential
Description: The user decides to join an existing platoon through a button press "Join" in the Graphical User Interface. This makes the vehicle join the platoon receiving and transmitting information amongst the platoon.
Cross References: Requirements: 1: 1.2, 1.4, 2

4: Use Case: Detect/Identify/track target

Actor: Radar
Type: Primary, essential
Description: The radar detects and identifies target information.
Cross References: Requirements: 1.1.1 , 1.1.2

5: Use Cases: Identify target, estimate speed, Backup Radar
Actor: Camera
Type: Primary, essential
Description: The camera either identifies the target vehicle, estimates the speed or acts as a backup for the radar
Cross References: Requirements: 1.1.1 , 1.1.2 , 1.4

6: Use Case: Adjust Speed
Actor: Vehicle Controller
Type: Primary, essential
Description: The Vehicle controller adjusts the speeds through the use of the brakes and throttles.
Cross References: Requirements: 1.3

7: Use Case: transmit
Actors: Radar, Camera, Lead Vehicle
Type: Primary, essential
Description: Information is transmitted by each of the actors.
Cross References: Requirements: 1, 2, 3, 4

8: Use Case: Send Data
Actor: Radio
Type: Primary, essential
Description: Sends information between vehicles in the platoon.
Cross References: Requirements: 1.4

9: Use Case: Send GPS Data, Broadcast Data
Actor: Lead Vehicle
Type: Primary, essential
Description: Sends and broadcast GPS data to the trailing vehicle.
Cross References: 1,2,3,4

10: Use Cases: Get Location, aid failed Radar
Actor: GPS
Type: Primary, essential
Description: Gets the location of a vehicle and acts as a helper during radar failure
Includes: Satellite Data
Cross References: Requirements: 1.2, 1.1.1

11: Use Cases: Satellite Data
Actor: External satellite
Type: Secondary
Description: Contains relevant GPS information of vehicles in the platoon.
Cross References: Requirements: 1.2

A high-level class diagram includes the following classes:

Element Name		Description
AccelerationMonitor	Monitors and implements the speed changes of items controlled by the CACC++System.	Brief description (e.g., purpose and scope).
Attributes		
	mph : Float	The speed of the items controlled by the CACC++System.
	accelerationMilesPerSecSqu : Float	The acceleration in (miles/seconds^2).
Operations		
	Brake()	This operation will decrease the speed of all the items in the CACC++System.
Relationships	VehicleSupervisory	The VehicleSupervisory notifies AccelerationMonitor if a speed change is necessary.
	ThrottleController	The acceleration of an item impacts the gas consumption of an item.

Element Name		Description
CACC++System		The overall system which is created when the CACC++ button is pressed. After the button is pressed, the system is activated.
Relationships	VehicleSupervisory	The VehicleSupervisory software makes the decisions that the CACC++ implements.
	Platoon	There can be multiple platoons using the CACC++ system at one time.
UML Extensions	Aggregation with obstacle.	

Element Name		Description
GPS		The GPS software that a car utilizes.
Attributes		
	location : Location	The location of a PlatoonCar.
UML Extensions	GPS has composition with PlatoonCar.	

Element Name		Description
GraphicalUserInterface		The screen that the user can utilize to direct the CACC++'s decisions

Attributes		
	message : String	The message that is displayed to the user.
Operations		
	DisplayWarning(Integer)	Displays the message associated with the warning inputted.
UML Extensions	GraphicalUserInterface has composition with PlatoonCar.	

Element Name		**Description**
LaneAssist		Controls the fuel flow of items.
Operations		
	Adjust()	This operation will adjust the lane of an item.
Relationships	*VehicleSupervisory*	The VehicleSupervisory notifies LaneAssist when a lane adjustment is necessary.

Element Name		**Description**
Obstacle		Anything the sensors or GPS locate.
Attributes		
	location : Location	The location of the obstacle.
	avoid : Boolean	Whether or not vehicles with CACC++ implemented should avoid the obstacle.
Relationships	Sensor	The sensors on should detect the obstacles in the path of a vehicle.
UML Extensions	PlatoonCar is derived from obstacle. Aggregation with CACC++System.	

Element Name		**Description**
PlatoonCar		A vehicle that is eligible to be part of a platoon (vehicles with the CACC++ system installed).
Attributes		
	speedInMPH : Float	The speed of the PlatoonCar.
	breakingDistance : Float	The distance that the PlatoonCar must have to break.
	followingDistance : Float	The distance that the PlatoonCar must be from the car in front of it.
Operations		

	AvoidObject()	This operation will tell the PlatoonCar if it should avoid some obstacle.
	ScanForPlatoon()	This operation will check for platoons in the area that the PlatoonCar can join.
	ListenForSignals()	This operation will listen to radio signals for information.
	SendSignals()	This operation will send radio signals to other PlatoonCars.
Relationships	*VehicleSupervisory*	The VehicleSupervisory is the software that commands the PlatoonCar.
	Radio	The radio transmits information to a PlatoonCar.
UML Extensions	PlatoonCar is derived from Obstacle. PlatoonCar has composition with Radio, Sensor, GPS, GraphicalUserInterface, and Platoon.	

Element Name		**Description**
Platoon		A collection of PlatoonCars.
Attributes		
	speedInMPH : Float	The speed of the PlatoonCars in the Platoon.
	maxSpeedInMPH : Float	The maximum legal speed for the cars in the Platoon.
	leadCar : PlatoonCar	The PlatoonCar at the front of the Platoon.
Operations		
	Split()	This operation will break the collection of PlatoonCars into multiple Platoons.
	Brake()	This operation will decrease the speed of all the PlatoonCars in the Platoon.
Relationships	*CACC++System*	There can be multiple platoons using the CACC++ system at one time.
UML Extensions	Platoon has composition with PlatoonCar.	

Element Name		**Description**
Radio		Allows transmission of information from one vehicle to another.
Relationships	PlatoonCar	The radio transmits information to a PlatoonCar.

UML Extensions	Radio has composition with PlatoonCar.	

Element Name		**Description**
Sensor		A sensor on a vehicle.
Operations		
	Warning()	An operation that detects objects.
Relationships	Obstacle	The sensors detect obstacles.
UML Extensions	Sensor has composition with PlatoonCar.	

Element Name		**Description**
ThrottleController		Controls the fuel flow of items.
Attributes		
	GallonsReleasing : Float	The amount of gas being released in gallons.
Operations		
	ReleaseGas()	This operation will release gas from an item.
Relationships	VehicleSupervisory	The VehicleSupervisory monitors the fuel consumption via the ThrottleController.
	AccelerationMonitor	The acceleration of an item impacts the gas consumption of an item.

Element Name		**Description**
VehicleSupervisory		The software that makes the decisions on how the CACC++System responds to situations.
Operations		
	InititateLaneCorrection()	This operation will notify the user of a PlatoonCar if their vehicle is changing lanes while the CACC++ is activated.
	InitiateSpeedCorrection()	This operation will decide if it is necessary to adjust the speed of all the PlatoonCars in the Platoon.
Relationships	CACC++System	The VehicleSupervisory software makes the decisions that the CACC++ implements.

	PlatoonCar	The VehicleSupervisory is the software that commands the PlatoonCar.
	LaneAssist	The VehicleSupervisory notifies LaneAssist when a lane adjustment is necessary.
	ThrottleController	The VehicleSupervisory monitors the fuel consumption via the ThrottleController.
	AccelerationMonitor	The VehicleSupervisory notifies AccelerationMonitor if a speed change is necessary.

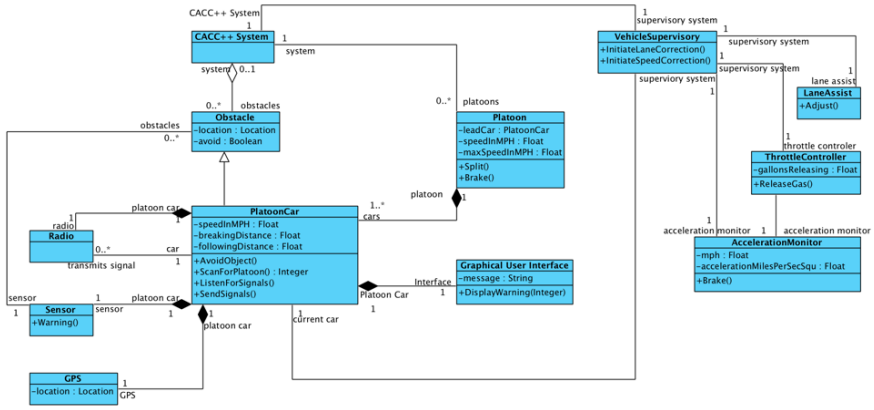

Figure 4: The high-level class diagram of the CACC++ system.

The scenarios of the system and their corresponding sequence diagrams are as follows:

Sequence diagrams are the interactions of different objects in time. They can show data structures involved in the scenario and communication between them. The full descriptions of the scenarios listed below can be found in section 5.2.

Scenario 1: The system is required to create a new task, but the memory is full. Since CACC++ does not use dynamic allocating memory the system must clear memory in a different way [6].

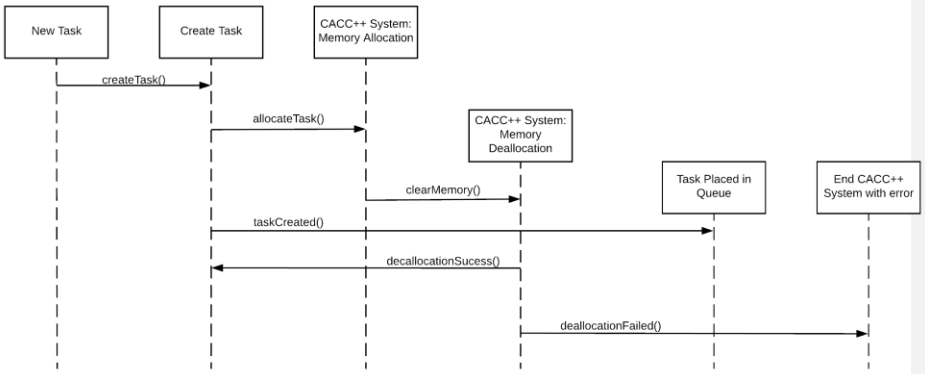

Figure 5: Sequence Diagram for Scenario 1

Scenario 2: The radio connection fails and before warning the user, the system should try to restart the radio. This will save time and improve the user experience [6].

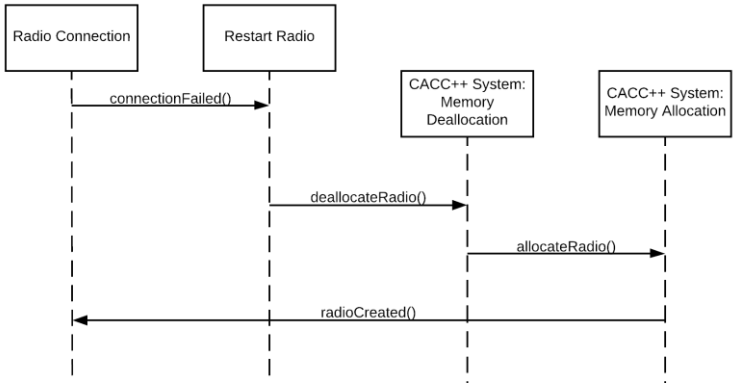

Figure 6: Sequence Diagram for Scenario 2

Scenario 3: The Acceleration monitor catches abnormal amounts of acceleration that may be dangerous to the user. The monitor catches the error in time and corrects the acceleration based on other data it has analyzed [6].

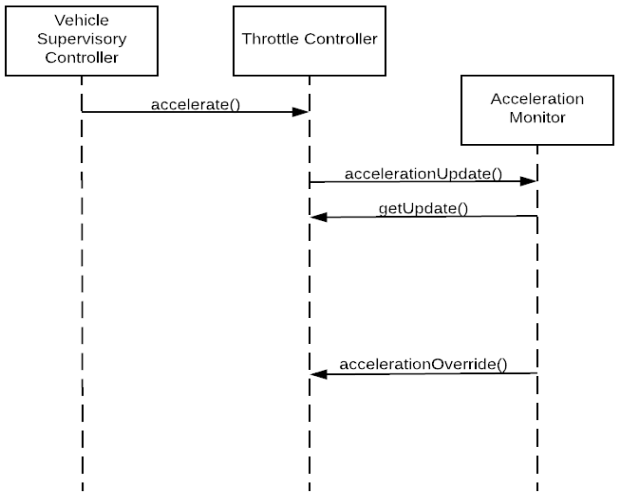

Figure 7: Sequence Diagram for Scenario 3

Scenario 4: Adverse road and weather conditions force the lead platoon car to make the decision to stop and get off the road. Is there a shoulder to pullover on, or does the platoon need to stop in an empty lane? The system if working properly, the lead vehicle should decide where to pull the platoon over [6].

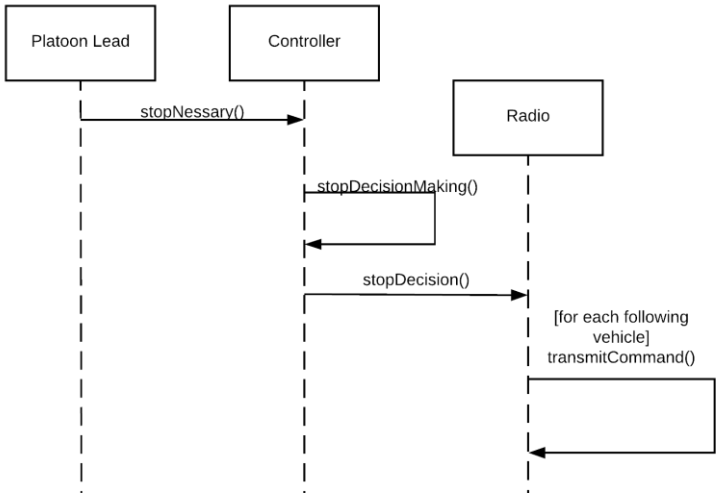

Figure 8: Sequence Diagram for Scenario 4

Scenario 5: During a turn the radar may be deceived into thinking that a vehicle is in its upcoming path [6].

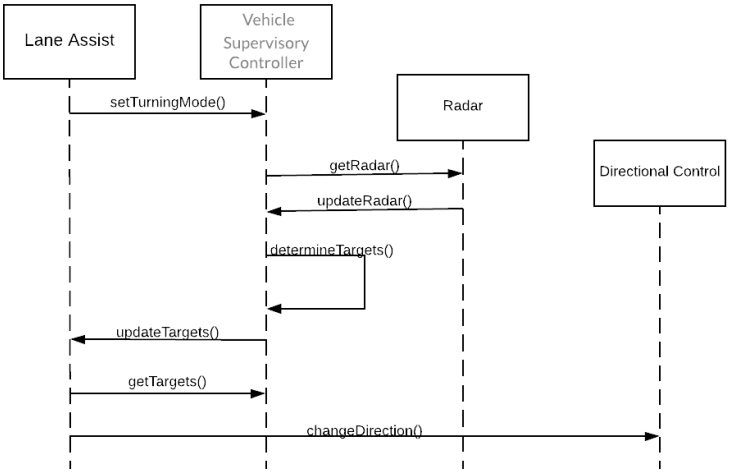

Figure 9: Sequence Diagram for Scenario 5

Scenario 6: The lead car in the platoon is beginning its accent over a large hill. To maintain speed the vehicle needs to accelerate. This means that the following cars also need to accelerate at the appropriate time [6].

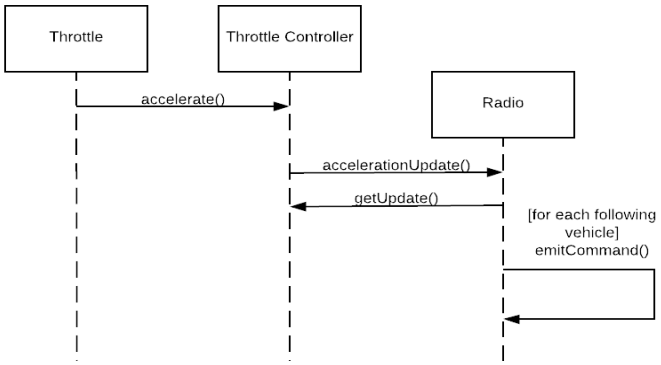

Figure 9: Sequence Diagram for Scenario 6

The state diagram of the CACC++:

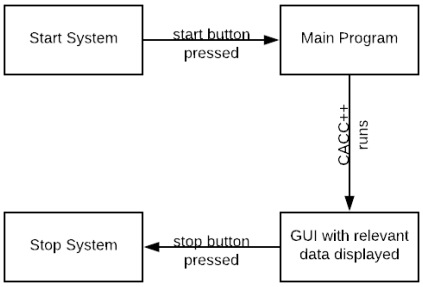

Figure 10: State Diagram for Scenario 6

The state diagram above displays all the changes that the user can make to the system while it is turned on. The System starts when the user presses the start button for the CACC++. This then begins the boot process for the main program. Once the CACC++ is successfully booted the user interface appears with relevant information for the user. Now the user has the choice to turn off the system, set a speed, or join a platoon. If the user decides to set a speed, they press the appropriate buttons, and after they release the button the system returns to the same user interface as previous. If they elect to join a platoon, they press the appropriate button, then are immediately prompted with a screen to set the follow distance. Once the follow distance has been set the system returns the user back to the same user interface with relevant data. The last option is to turn off the system. This can be done by simply pressing the appropriate button. The process then ends.

5 Prototype

The prototype will display a system that will be able to safely assist a driver in maintaining control over their vehicle. It will be able to function accordingly, and in a safe and expected manner, in the six scenarios defined below. The user will have access to an interface that will show the communication between the system and the user. In this interface, the system will display all automatic decisions as well as required and expected input from the driver.

5.1 How to Run the Prototype

The prototype will be accessible on our website and through a web browser a user will be able to run the system. The interface of our prototype will explore the six scenarios previously mentioned and will be able to properly depict these scenarios, and show the user how the system makes adjustments automatically. By the creation of the first prototype the system will provide a basic interface for the user to understand while testing the prototype. The final prototype will have all required functionality of the system and will be able to act autonomously in making decisions to assist a driver in the required scenarios.

5.2 Sample Scenarios

Using the prototype, the user will be able to run through six main scenarios to test the system's functionality. Each scenario will challenge the system in a different manner in order to ensure that it has all required functionality. These scenarios are described as follows.

In Scenario 1 a vehicle is in a platoon of two and is the trailing vehicle. Everything is working as expected and there is a task in the vehicle controller that cannot start due to the lack of memory. While dynamically allocated memory is not used, each instance of a task starting, procedure called, or object created does require memory to be allocated by the operating system (OS). The vehicle controller (VC) creates individual objects to track targets. One of the targets is the lead vehicle; others are vehicles that are not part of the platoon. These objects will likely be created to detect failure to start tasks. The system should be able to diagnose a lack of memory and decide how to recover.

For Scenario 2 a vehicle is in a platoon of two and is the trailing vehicle. Everything is working according to plan when the radio system no longer responds. Assume the radio system also has some dynamic object creation/destruction to track communication links to multiple vehicles and to the infrastructure. If a failed garbage collection scheme occurred, the radio system should be restarted by a command from the VC.

Scenario 3 has a coding error in the arbitration logic within the vehicle supervisory controller, not caught during system verification and testing. This situation results in a large acceleration command to the throttle controller that is inappropriate given the current system context. An independent monitoring function detects the discrepancy between the commanded value to the throttle controller and the system context based on system inputs and state variables, and determines that the throttle command must be reduced to a more moderate value.

In Scenario 4 the lead vehicle in the platoon determines an emergency stop is required due to road conditions. Based on current state of the platoon and the vehicle performance envelopes the platoon decides if it should stop or divert its members to adjacent empty lanes or shoulder.

Scenario 5 explores when traffic in adjacent lanes can appear to be in the anticipated path of the lead vehicle. This is especially true for large trucks which have a large radar signature. The system must discriminate real targets from false targets when more than one moving vehicle is detected.

Lastly, Scenario 6 tests when traveling in a platoon in moderately hilly terrain vehicles need to provide additional torque to maintain speed and separation. The system must ensure that all members of the platoon maintain speed and separation when going up or down a hill.

In each of these scenarios our prototype will display to the user any changes in their vehicles speed as well as the speed of the vehicles in the platoon. Warning messages, speed updates, platoon status, and system control access will all be displayed to the user, and available on the user's steering wheel and dashboard. All vehicles in the platoon will be in communication with each other internally and will relay relevant information to their drivers.

6 References

Start of your text.

[1] D. Thakore and S. Biswas, "Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks," Proceedings of IEEE Military Communication, Atlantic City, October 2005.

[2] Cooperative Adaptive Cruise Control++ (CACC++) [Online] / auth. Milam Mr. William. - http://www.cse.msu.edu/~cse435/Projects/F2018/ProjectDescriptions/cacc.pdf

[3] Introductory guide to sensors: https://www.keyence.com/ss/products/sensor/sensorbasics/vision/info/

[4] Actuator definition : Car manufacturing [online] / https://www.actuatorzone.com/blog/automotive-industry/linear-actuators-automotive-industry/

[5] K. E. Wiegers, "SRS Template IEEE," p. 8, 1999.

[6] "The Project." *Team CACC++1*, cse.msu.edu/~sydlikb1/.

7 Point of Contact

For further information regarding this document and project, please contact **Dr. James E. Daly** at Michigan State University (dalyjame at cse.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.