Software Requirements Specification (SRS)

Cooperative Adaptive Cruise Control++

Authors: Hayden Cederstrom, Jessica Clappison, Joey Baum, Andrew Morrissette, Jeff Fallon

Customer: Vehicle Manufacturers

Instructor: Dr. James Daly

1 Introduction

- Provide an overview of the entire SRS subsections
- Indicate the topics that will be covered in this document.

This document contains the Software Requirements Specification (SRS) for the Cooperative Adaptive Cruise Control++ (CACC++) project. The document is broken up into six sections. Section 1 is the Introduction to the SRS document, which is broken up into four sections: purpose, scope, definitions, acronyms, and abbreviations, and organization. Section 2 will contain an overall description of the software product, and it is broken up into six sections: product perspective, product functions, user characteristics, constraints, assumptions and dependencies, and apportioning of requirements. Section 3 contains specific requirements. Section 4 contains modeling requirements. Section 5 describes the prototype, and is broken up into two sections: how to run the prototype and sample scenarios. Section 6 will contain references.

1.1 Purpose
What’s the purpose of the SRS document?

Specify the intended audience.

The purpose of this SRS document is to specify the requirements for the software that will be required to create the CACC++ project. It will describe the software product’s intended functionality, characteristics, dependencies, and constraints. It will also describe the functionality of the prototype and how to use it.

The intended audience of this SRS document includes software developers who will create the software for the CACC++ project, as well as project managers and vehicle manufacturers.

1.2 Scope

- Identify SW product(s) to be produced by name
- Describe the application of SW being specified, including benefits, objectives, goals. What is the application domain? (e.g., embedded system for automotive systems, graphical modeling utility) This is the domain description of the application.
- Explain what SW product will, and if necessary, will not do. This is the requirement of the application.
- Be consistent with similar statements in higher-level specifications (e.g., the original project specification from customer)

The products in this document are listed as follows: Cooperative Adaptive Cruise Control++, Vehicle Control System, Forward Looking Camera, Radar Sensing, Radio Sensing, GPS System, Electronic Throttle Control, and Brake By Wire.
The benefit of this system will be that it autonomously controls the vehicle with little input needed from the driver. The objective is to create a system of autonomous vehicles that can use their sensors to communicate with each other and create a safer environment. The main goal is to eliminate vehicle collisions and accidents that occur from human error. Our software is an embedded system for automotive systems.

This system will be able to maintain a constant forward vehicle speed that is specified by the driver. The system will be able to use forward radar and camera sensors to detect when another vehicle is in its path, and it will be able to adjust the speed of the vehicle accordingly using throttle and braking control. It will also be able to communicate this information with the vehicles around it using its GPS system. Using the information from the GPS system, a platoon of vehicles will be created that follows a lead vehicle. The braking limit is set at 2G, and the vehicle will only be able to decelerate at 2G. The system will also be able to use adaptive cruise control, lane keeping, curve speed assist, and hill management to apply throttle and braking at different rates under different conditions.

1.3 Definitions, acronyms, and abbreviations

- Define all terms, acronyms, and abbreviations need to understand the SRS. If this section is extensive, then move to an appendix. It is also possible to provide a link to other resources for extensive terminology explanation.

SRS - Software Requirements Specification

CACC++ - Cooperative Adaptive Cruise Control++

GPS - Global Positioning System

Ref - Reference

MPH - Miles Per Hour
1.4 **Organization**

- Describe what the rest of the SRS contains
- Give the organizational structure of the SRS.

Section 1 is the Introduction to the SRS document, which is broken up into four sections. Section 1.1 is about the purpose of the SRS document and its intended audience. Section 1.2 is about the scope of the project. The scope covers the software products to be produced by name. The scope also includes a description of the application of software being specified, which includes benefits, objectives, and goals. The scope will include the application domain and the domain description. It will also explain the functionality and lack of functionality of the software. Section 1.3 will include definitions of terms, acronyms, and abbreviations used in the SRS. Section 1.4 will describe the content of the rest of the SRS, as well as the organizational structure of the SRS.

Section 2 will contain an overall description of the software product, and it is broken up into six sections. Section 2.1 will give a product perspective. It will give context for the product, as well as a diagram of how our product fits within a vehicle. Section 2.1 will also describe the interface constraints of our system. Section 2.2 will describe the product functions. This will be a summary of the functions that the software will perform, and will contain a goal diagram for the
system. Section 2.3 will describe user characteristics, including background, skill level, and general expertise. Section 2.4 describes the constraints, and will include safety-critical properties, and properties that will cause the system to not perform properly. Section 2.5 will describe the assumptions made about the hardware, software, environment, and user interactions, as well as dependencies. Section 2.6 will describe requirements determined based on negotiations with customers, which will be beyond the scope of the current project.

Section 3 describes specific requirements. It will contain an enumerated list of requirements.

Section 4 describes modeling requirements. It specifies the bridge between the application domain and the machine domain. It also describes the notation of each diagram introduced in this document. It will contain class diagrams, descriptions of example scenarios, and a state diagram for all key classes that participate in the example scenarios.

Section 5 will describe the prototype, and the functionality that it will have. Section 5 is broken up into two sections. Section 5.1 will describe how to run the prototype, including system configurations, plugins, and OS or networking constraints. It will contain the URL for the prototype. Section 5.2 will describe sample scenarios using real data and problem scenarios. It will include screen captures of what the prototype will produce.

Section 6 will describe references. It will include a list of all documents referenced in this document, including title, report number, date and publishing organization.

2 Overall Description

- Give a brief introduction of what information will be covered in this section.

This section will discuss the context of the CACC++ software product as well as the functions that it will possess. It will describe what type of user will be using the software and what constraints the software will have. Finally, this section will cover the assumptions and
dependencies that the software will have along with requirements that have been deemed to be beyond the scope of the current project.

## 2.1 Product Perspective

- **Describe the context for the product**
- **Is it one element that is part of a bigger system? If so, then give a pictorial representation or diagram (e.g., data flow diagram – DFD, block diagram) that describes how your product fits.**
- **Interface Constraints:**
  - System interfaces
  - User interfaces
  - HW interfaces
  - SW interfaces
  - Communication interfaces
- **Other types of constraints:**
  - Memory
  - Operations
  - Site adaptation operations (customization that is done on-site).

The automotive industry has been increasingly using new technology to enhance safety features for people while they drive. Drivers are not perfect and are increasingly needing assistance from their vehicles to help keep them stay safe and satisfied. One major part of driving is cruising behind other vehicles. This act seems simple at first, but has many potential complications when looked at closer. An increase in distractions and higher vehicle speeds make it so a driver’s focus is not enough to prevent accidents. If a driver reaches down for their
phone for a second at 70 miles per hour and the car in front of them slams on their breaks, it will result in a horrible event that the car would not be able to stop. However, with the CACC++ software system it will be able to prevent devastating accidents by controlling the car and calculating a safe following distance for the driver. The CACC++ software system will communicate with other vehicles and pass along important information to help calculate what the vehicle should do.

The CACC++ software system is one element of the entire cars control system. Below is a representation of where this piece of software would be placed within the cars control system.

Some constraints that the CACC++ will have are due to the limits of how computers communicate with one another. The user interface will have to used by someone who is able to see. Hardware and software interfaces are limited by weather and computational power respectively. The system interface needs power to communicate with other systems.

2.2 Product Functions
The major functions for CACC++ software will perform many tasks that will help keep the driver safe. One major function will use the car's different sensors and calculate if the vehicle should slow down or speed up depending on what is occurring in front of it. Another function will be in charge of making sure that the vehicle will be centered in its lane. Other functions will tend to curve speed assist and calculating the speed when going up and down hills. Safety functions will have to be put in place to make sure things are functioning correctly when the system is running. Below is a high-level goal diagram for the CACC++ system that has nodes as boxes that are linked together by relationships (lines).
2.3 User Characteristics

- Expectations about the user (e.g., background, skill level, general expertise)

Many different users will be able to use the CACC++ because driving is a major part with developing societies. Younger users might not take full advantage of the CACC++ because they are new to driving and don’t know how it could be used to help them. More skilled and older drivers who have driven a lot would tend to use the CACC++ system more because it is a convenience to them. There would be other potential users who might not trust the system fully enough to use the system continuously.

2.4 Constraints

- See list of possible constraints from IEEE SRS document.
- Give English descriptions of safety-critical properties
- Give English descriptions of other properties that if violated, the system will not perform properly.

Safety-critical systems consist of the potential for death to occur, severe damage or obstruction to property, and environmental harm. The CACC++ has the potential to have these events take place if a malfunction occurs.

2.5 Assumptions and Dependencies

- Assumptions made about the HW, SW, environment, user interactions.
The vast majority of sections of this system assume that the car is fully functional. It also assumes it has the capacity to successfully connect to GPS as well as send information to cars in a platoon. Furthermore, the system assumes that all the cars in a given platoon use the CACC++ system.

The GPS system is dependant on GPS satellite system working. If the system is unable to get information from other vehicles through GPS it will not be able to function correctly. The location then impacts where the CACC++ system may be used.

2.6 Approportioning of Requirements

- Based on negotiations with customers, requirements that are determined to be beyond the scope of the current project and may be addressed in future versions/releases.

No requirements have been determined as beyond the scope of the current project. Features like center lane assist are and hill management are expected to be implemented with the adaptive cruise control system.

3 Specific Requirements

- Give an enumerated list of requirements.
- As appropriate, use a hierarchical numbering scheme.

1. Sample requirement at the top level
   1.1. Level 2 requirement example
   1.2. Another Level 2 requirement
2. Select the “Requirement” Style.

---------------------------------------------------------------------------------------------------

3.1 External Interface Requirements

3.2 Functional Requirements

3.2.1 Radar Sensing

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Radar Sensing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The Radar Sensing detects, id and tracks a target vehicle.</td>
</tr>
<tr>
<td>Trigger</td>
<td>A car is detected in the radar</td>
</tr>
</tbody>
</table>
| Basic Path    | 1. Affixed to the front of the car  
               | 2. Used to detect, id and track vehicles in front of the CACC++ vehicle  
               | 3. Information used to determine best vehicle operations |
| Postcondition | Information is collected from the vehicle in front of the vehicle. |
| Exception Paths | There is no car in front of the vehicle. |

3.2.2 Radio Sensing

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Radio Sensing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The Radio Sensing communicates with target vehicle, trailing vehicle and communicates with infrastructure.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Information is received from the Radar</td>
</tr>
</tbody>
</table>
### Basic Path

| Basic Path | 1. Passes along information received from the Radar Sensing to the car behind  
|            | 2. Receives information from the car preceding it  
|            | 3. Communicates with the system architecture |

### Postcondition

Vehicle speed, id and position is passed along to trailing car. The same information from the car in front is received. Information also passed to Vehicle Controller.

### Exception Paths

The car is not in a platoon

### 3.2.3 Electronic Throttle Control

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Electronic Throttle Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The Electronic Throttle Control regulates vehicle speed to commanded speed by adding/removing power.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Receives new information from Camera, Radar and GPS</td>
</tr>
</tbody>
</table>

| Basic Path | 1. The Throttle Control receives new information  
|            | 2. The car speed is adjusted based off information via power |

### Postcondition

The car speed is adjusted as needed.

### Exception Paths

The car is not in a platoon.

### 3.2.4 Brake By Wire

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Brake by Wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The Brake by Wire regulates vehicle speed by applying brakes to bring vehicle speed to commanded speed.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Receives new information from Camera, Radar and GPS</td>
</tr>
</tbody>
</table>
### Basic Path

<table>
<thead>
<tr>
<th>Postcondition</th>
<th>The car speed is adjusted as needed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception Paths</td>
<td>The car is not in a platoon.</td>
</tr>
</tbody>
</table>

### 3.2.5 GPS System

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>GPS System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The GPS System maintains accurate vehicle position, speed and direction information. Also aids radar system in maintaining distance and differentiating between vehicle targets and known targets.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The GPS system is active as long as the system is functional</td>
</tr>
</tbody>
</table>
| Basic Path    | 1. Consistently keeps track of current vehicle speed, location and direction  
                2. Passes along info to car behind it.  
                3. Uses information to aid radar |
| Postcondition | The GPS information is passed to cars and to vehicle controller. |
| Exception Paths | None |

### 3.2.6 Forward Looking Camera

| Use Case Name | Forward Looking Camera |
### Description
The Forward Looking Camera visually identifies target vehicle and estimates distance and relative speed. Also a backup for radar system.

### Trigger
A car is detected in the camera view.

### Basic Path
1. Detects leading car’s position  
2. Passes collected information to radar system

### Postcondition
Camera information is passed along to be used to adjust car speed.

### Exception Paths
No car is present in the camera’s view

---

3.2.7 Vehicle Controller

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Vehicle Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The Vehicle Controller coordinates all subsystems of the CACC++</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>The Vehicle Controller is constantly using information.</td>
</tr>
</tbody>
</table>
| **Basic Path**    | 1. Detects vehicle speed, speed of lead vehicle and adjusts to maintain safe distance  
2. Maintains state of vehicle and operating environment information  
3. Commands throttle and brakes  
4. Receives information from radar system  
5. Sends and receives information from radio system |
| **Postcondition** | The car’s speed is position is adjusted if need be. Rest of system passes along information. |
| **Exception Paths** | None |
4 Modeling Requirements

Use-Cases:

- This is the specification portion of the requirements document. (Specifying the bridge between the application domain and the machine domain.)
- For each new diagram type introduced, describe the notation.
  - Give and describe use case diagrams
  - Use the template below to describe each use case.

  - Each goal may be satisfied by 1 or more use cases
  - Each use case should refer to 1 or more requirements (in Section 3)
  - Give and describe a high-level class diagram that depicts the key elements of the system
  - Include a data dictionary to describe each class, its attributes, its operations, and relationships between classes.

- Representative Scenarios of System:
  - Give English descriptions of representative scenarios for each use cases.

§ Check: use instances of the class names from class diagram; refer to the terms used in use case diagram

§ For each scenario, give a corresponding sequence diagram

§ Check: Objects should be instances of classes in class diagram
Create and explain a state diagram for all key classes that participate in the scenarios (from above).

- § Check: that all scenarios can be validated against the state diagrams.
- § Check that the events, actions are modeled in the class diagram.
- § Check that all variables referenced in the diagrams are declared as attributes in the class diagram.

**High-Level Class Diagram**

The High-Level Class Diagram is used to see how the entire system interacts with each other through user input, and what get information from other classes. Each Circle is a Use Case that does something in the system. The lines indicate what system includes different parts, ie. Radar and Radio use the Forward Looking Camera.
Use Case Diagram

The Use Cases depict an action being done on the system through certain classes and how they are enacted.

Use Case: Turn On
Actors: Vehicle Operator
Type: Primary and Essential
Description: The Vehicle Operator turns the CACC++ On. Once powered up the system will actively monitor its surroundings and adjust speed accordingly.
Use-Cases: Perform Adjust Speed per Vehicle Control

Use Case: Turn Off
Actors: System Initiated and Vehicle Operator Initiated
Type: Primary and Essential.
Description: If any errors occur, A sudden change in traffic or the owner decides to turn off the system the system will Power Off.

Use Case: Vehicle Controller
Actors: System Initiated
Type: Primary and Essential
Description: Actively uses all subsystems to set whether the vehicle needs to speed up or slow down, as well as if the system should turn off.
Use-Cases: GPS System, Electronic Throttle, Radar Sensing, Radio Sensing, Forward Looking Camera, Brake by Wire
**Use Case: GPS System**

Actors: System Initiated  
Type: Primary and Essential.  
Description: The GPS System maintains accurate vehicle position, speed and direction information. Also aids radar system in maintaining distance and differentiating between vehicle targets and known target.  
Cross Ref: Vehicle Controller

**Use Case: Electronic Throttle**

Actors: System Initiated  
Type: Primary and Essential  
Description: Turns on via the Vehicle Controller to adjust speed of the car.  
Use-Cases: Vehicle Controller

**Use Case: Radar Sensing**

Actors: System Initiated  
Type: Primary and Essential.  
Description: Radar Sensing detects, id and tracks a target vehicle.  
Cross Ref: Vehicle Controller  
Use-Cases: Forward Looking Camera

**Use Case: Radio Sensing**

Actors: System Initiated
Type: Secondary and Essential.
Description: The Radio Sensing communicates with target vehicle, trailing vehicle and communicates with infrastructure.
Cross Ref: Vehicle Controller, GPS System
Use-Cases: Forward Looking Camera

Use Case: Forward Looking Camera
Actors: System Initiated
Type: Primary and Essential.
Description: The Forward Looking Camera visually identifies target vehicle and estimates distance and relative speed. Also a backup for radar system.
Cross Ref: Radar Sensing, Radio Sensing

Use Case: Brake by Wire
Actors: System Initiated
Type: Primary and Essential.
Description: The Brake by Wire regulates vehicle speed by applying brakes to bring vehicle speed to commanded speed.
Cross Ref: Vehicle Control, GPS System
State Diagrams

State Diagrams are used to show each Class and their states they can be in. A Square is the state they are currently in and the arrow’s indicate the transition from one state to another. The first state is indicated by the state attached to the black circle.
# Dictionaries

Dictionaries are used to help aid the developers creating each class and helps describe what each function does within each class.

<table>
<thead>
<tr>
<th>Vehicle Controller</th>
<th>Controls all subsystems and actively makes decisions based on its surrounding.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>CurrentState:VehicleOBJ</td>
<td>Current State of the Vehicle</td>
</tr>
<tr>
<td>FrontCarState:VehicleOBJ</td>
<td>Current State of the Front Car (If there is one).</td>
</tr>
<tr>
<td>TrailingCarState:VehicleOBJ</td>
<td>Current State of the Trailing Car (If there is one).</td>
</tr>
<tr>
<td><strong>Operations</strong></td>
<td></td>
</tr>
<tr>
<td>CalculateDecision()</td>
<td>Check all systems and determine if car needs to slow down or speed up.</td>
</tr>
<tr>
<td>Return string</td>
<td></td>
</tr>
<tr>
<td>isSlow(string)</td>
<td>If Decision returned “slow”, meaning the current car is too slow, the system will speed up.</td>
</tr>
<tr>
<td>Return Bool</td>
<td></td>
</tr>
<tr>
<td>isFast(string)</td>
<td>If Decision returned “fast”, meaning the current car is too fast, the system will slow down.</td>
</tr>
<tr>
<td>Return Bool</td>
<td></td>
</tr>
<tr>
<td>saveState(int,int)</td>
<td>Save the current State (current speed and GPS Location) of the vehicle.</td>
</tr>
<tr>
<td>Return vehicleOBJ</td>
<td></td>
</tr>
</tbody>
</table>
### Relationships

<table>
<thead>
<tr>
<th>SendInformation(vehicleOBJ)</th>
<th>Send current state to Radio.</th>
</tr>
</thead>
</table>

### UML Extensions

- **Forward Looking Camera**
  - **Visually identifies the target vehicle and estimates distance and relative speed.**
  - **Is a backup for radar for tracking the target vehicle.**

#### Attributes

| TargetVehicle:VehicleOBJ | A vehicle object describing the target vehicle. |

#### Operations

<table>
<thead>
<tr>
<th>isCar() Return Bool</th>
<th>Checks if object in view is a vehicle.</th>
</tr>
</thead>
<tbody>
<tr>
<td>isLeading() Return Bool</td>
<td>Checks if vehicle detected is the leading vehicle in the platoon.</td>
</tr>
<tr>
<td>AssignID() Return int</td>
<td>Assigns an identification number to the vehicle.</td>
</tr>
<tr>
<td>calculateDistance() Return float</td>
<td>Calculates distance to the target vehicle.</td>
</tr>
<tr>
<td>CreateVehicleOBJ() Return vehicleOBJ</td>
<td>Returns a vehicle obj to the radar</td>
</tr>
</tbody>
</table>

### Relationships

<table>
<thead>
<tr>
<th>Communicates with Radar</th>
</tr>
</thead>
</table>
### Radar Sensing

<table>
<thead>
<tr>
<th>Attributes</th>
<th>TargetVehicle:VehicleOBJ</th>
<th>Vehicle object describing the sensed vehicle.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>carDetected() Return Bool</td>
<td>Determines if a vehicle has been detected</td>
</tr>
<tr>
<td></td>
<td>createVehicleOBJ() Return Bool</td>
<td>Creates a vehicle object and returns to the radio.</td>
</tr>
</tbody>
</table>

### Relationships

Radar Sensing communicates with Radio and Vehicle Controller

### Radio Communication

<table>
<thead>
<tr>
<th>Attributes</th>
<th>vehicleOBJ:VehicleOBJ</th>
<th>Vehicle object to be added</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>to the platoon.</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>getNotification()</td>
<td>Receives a notification from the radar sensor, returns to checkGPS()</td>
<td></td>
</tr>
<tr>
<td>checkGPS()</td>
<td>Sends vehicleOBJ to GPS to put in platoon.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relationships</th>
<th>Radio sensing communicates with GPS System and Vehicle Controller</th>
</tr>
</thead>
</table>

| UML Extensions                        |                                                                  |
|----------------------------------------|                                                                  |

<table>
<thead>
<tr>
<th>GPS System</th>
<th>Maintains accurate vehicle position, speed and direction information. Aids radar system in differentiating between vehicle targets and known fixed targets. Aids vehicle to maintain distances when radar fails.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>vehiclePosition:String</td>
<td>Current Position of the Vehicle</td>
</tr>
<tr>
<td>vehicleSpeed: Float</td>
<td>Current Speed of the vehicle</td>
</tr>
<tr>
<td>vehicleDestination: String</td>
<td>Destination of the vehicle</td>
</tr>
<tr>
<td>Operations</td>
<td>CalculatePosition() Return string</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td></td>
<td>calculateSpeed() Return float</td>
</tr>
<tr>
<td></td>
<td>calculateDirections() Return string</td>
</tr>
<tr>
<td></td>
<td>differentiateOBJ(vehicleOBJ) Return Bool</td>
</tr>
<tr>
<td></td>
<td>maintainDistance(vehicleOBJ) Return float</td>
</tr>
<tr>
<td>Relationships</td>
<td>GPS communicates with Radio Sensing and Vehicle Controller</td>
</tr>
<tr>
<td>UML Extensions</td>
<td></td>
</tr>
</tbody>
</table>

<p>| Vehicle Controller | Controls all subsystems and actively makes decisions based on its surrounding. | |
| Attributes | | |
| CurrentState:VehicleOBJ | Current State of the Vehicle | |
| FrontCarState:VehicleOBJ | Current State of the Front Car (If there is one). | |</p>
<table>
<thead>
<tr>
<th><strong>TrailingCarState:</strong> VehicleOBJ</th>
<th>Current State of the Trailing Car (If there is one).</th>
</tr>
</thead>
</table>

**Operations**

<table>
<thead>
<tr>
<th><strong>CalculateDecision()</strong></th>
<th>Check all systems and determine if car needs to slow down or speed up.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>isSlow(string)</strong></td>
<td>If Decision returned “slow”, meaning the current car is too slow, the system will speed up.</td>
</tr>
<tr>
<td><strong>isFast(string)</strong></td>
<td>If Decision returned “fast”, meaning the current car is too fast, the system will slow down.</td>
</tr>
<tr>
<td><strong>saveState(int,int)</strong></td>
<td>Save the current State (current speed and GPS Location) of the vehicle.</td>
</tr>
<tr>
<td><strong>SendInformation(vehicleOBJ)</strong></td>
<td>Send current state to Radio.</td>
</tr>
</tbody>
</table>

**Relationships**

Vehicle Control directly uses every other system to dynamically interact with each other.

**UML Extensions**

**Brake Wire**

Regulates vehicle speed by applying brakes to bring vehicle speed to commanded speed.

**Attributes**

| currentSpeed: float | Current speed of the Vehicle |

**Operations**

| applyBrakes() | Applies brakes to slow |
### Return void

- **stopBrakes()**
  - Return void
  - Stop applying brakes to speed up the vehicle

### Relationships

- Brake by wire communicates with the Vehicle Controller

### UML Extensions

- Electronic Throttle Control
  - **Objective:** Regulate vehicle speed to commanded speed by adding/removing power

<table>
<thead>
<tr>
<th>Attributes</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>currentSpeed: float</td>
<td>Current speed of the Vehicle</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operations</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>applyPower()</td>
<td>Return void</td>
<td>Applies power to speed up the vehicle.</td>
</tr>
<tr>
<td>stopPower()</td>
<td>Return void</td>
<td>Stops power to slow down the vehicle</td>
</tr>
</tbody>
</table>
### Scenario

A common scenario would be a User turning on the CACC++ System, there is a car in front of the vehicle and behind, and the car in front is very far away and the car behind the vehicle is very close. The Vehicle Controller will sense this by checking all of the other sub-systems and will decide to turn on the gas to obtain a safer distance between both cars.

### Sequence Diagram

The Sequence Diagram is to demonstrate the timing and interactions if the above scenario happened. White boxes with text indicate a class, arrows indicate call and response functions between the different classes, a blank white box indicates time spent retrieving a response from a function in another class.
5 Prototype

- Describe what your prototype will show in terms of system functionality.

5.1 How to Run Prototype

- Describe what is needed to run your prototype
- What system configuration? (Should be accessible through web.) Are there plugins? Are there any OS or networking constraints. Give the URL for the prototype.
Prototype v1 does not have to executable per se. But there should be sufficient number of interfaces for the customer to understand the development's interpretation of the requirements.

Prototype V2 should also be accessible via a webpage. It should be executable and provide an interactive interface.

To run the prototype, an interactive interface will need to be created. This interface will include several input boxes and an output box. The input boxes will represent the different sensors receiving different information from the system. Users will input information, such as how fast the car in front is travelling, and the prototype will do calculations to determine how the car will react to that information. This information, such as if the car needs to accelerate or decelerate, will be displayed in the output box. All of this will be able to be done in browser in the prototype page of the website. The calculations and changes to the visual of the prototype will all be done in javascript. There will be no additional plugins or OS constraints.

5.2 Sample Scenarios

Give a sample scenario of using your system. Use real data and problem scenarios. Include screen captures illustrating what your prototype produces. As always, be sure to describe all figures.

A sample scenario for the prototype is demonstrated by inputting different speeds into the input boxes. For example, simulation of the the leading car shifting in speed could be done by having the leading car have a start speed of 20mph to an end speed of 50mph. This information would be inputted into the input boxes and then processed through the prototypes javascript code. It would use internal calculations to determine
how the system would react to this situation. The result of this situation would be shown
in the output box and would show the shift in speed that the car needs to go through to
handle this situation. All of this will be done on this page:

http://www.cse.msu.edu/~baumjose/prototype.html

6 References

· Provide list of all documents referenced in the SRS
· Identify each document by title, report number, date, and publishing organization.
· Specify the sources from which the references can be obtained.
· Include an entry for your project website.

Start of your text.

Intermittently Connected Wireless Networks,” Proceedings of IEEE Military
Communication, Atlantic City, October 2005.

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 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.