Cruise control

Software document (prototype)

V.0.1.3

Team Name: Team White Hat

Team Members: Christopher Cousillas, Matthew Jacobson, Justin Kroeger, Lauren Melendez

Pledge: I pledge my honor that I have abided by the Stevens Honor System - Team White Hat

Cruise Control Prototype

1. <u>Table of contents</u>

Executive Summary

Cruise control is a system that controls the speed of a vehicle. This system can be through software or through hardware. A cruise control system's main purpose is to maintain the speed of the vehicle and control the throttle of the vehicle in order to maintain this speed. As a driver, safety is my priority. This cruise control allows the user to be safe, along with pedestrians and other drivers. When going on long drives, cruise control is the perfect way to avoid fatigue and allows the user to be more comfortable.

Introduction

Customer Needs

This cruise control system is designed as a convenience for users who may be driving for long periods of time or would like assistance in maintaining a consistent speed. To this point, the system is entirely designed with both convenience for the user in mind, along with a high reliability, being that the system is controlling a motor vehicle. At a high level, the system will engage via input from the user, and will maintain a determined speed, such that the user does not need to keep their foot on the pedal. This will help users maintain a higher comfortability when driving for long periods of time, along with keeping them from subconsciously raising their speed above the speed limit.

Features

In order to create a system that meets the users needs, a series of features that allows this system to function as a whole shall be specified, in order to maximize the user experience. To initially turn the system on, a switch shall be implemented that starts the cruise control. It should be noted, however, that this system shall not have the ability to activate at low speeds, for safety reasons. Likewise, there shall be an ability to deactivate the system. This will be implemented via the brake. When the user applies the break, the cruise control will deactivate and the user shall reapply the gas pedal to continue driving. While the system is activated, it shall be able to measure the current speed of the vehicle, along with maintaining a set speed, via an interface that can connect the software to the actuators in the vehicle. The consumer shall also be able to change the cruise control speed. This will be done with an interface (i.e a button) that can set the speed that the cruise control should maintain. Finally, the system shall have a graceful shutdown and startup. When the car is shut down, the system shall shut down and start up in a manner that is safe, and unharmful to the vehicle, or the cruise control system. These features, when implemented in a persistent system, will create a user experience that is consistent, safe, and intuitive.

Reliability

In a system such as this, where the functionality of it directly impacts the safety of the user, along with other drivers on the road, an immense reliability is absolutely imperative. In order to meet these reliability demands, this system will maintain a 4-9 reliability. This means that the reliability is 99.99%, so that you can be sure that you can trust the system when in use. The system will maintain power with the car, so the user can be sure that the system will never lose power, and deactivate unexpectedly. The reliability of this system shall be tested extremely thoroughly, as to be sure that it behaves exactly as is expected, maintaining a 4-9 reliability

Performance

This system shall perform, at a high level, and in real time. The time the system needs to activate after receiving input shall be kept to a minimum, and in this system, it will be 100ms. Although this system will not have adaptive cruise control, it is important to maintain good design, such that a potential upgrade could be applied to implement such a feature. It is not only important to us that we create a design that our clients enjoy using, we strive to also create something that they trust.

Project Requirements

Elicitation Work Products

This cruise control system is vital to the users' safety and shall function exactly as described. The system shall maintain a speed set by the user. It shall maintain this speed in various different conditions, such as moving up or down hills. The system shall only activate if it is deemed safe to utilize. If the vehicle is moving too fast, the user does not have the seat belt on, or if the tire pressure is too low, the system will not activate. This system can be developed by our current team of four, with the use of an agile method of development. Our team will develop the software controlling the cruise control, creating the logic behind the system. On the technical side, our team will be using GitLab for version control. This software allows our team to effectively collaborate, and make the most out of our agile process. The requirements of this system were created by our team along with influence from Professor Peyrovian. These requirements are exhaustive and cover all scenarios. The requirements decided upon are as followed :

Functional Requirements:

1.1) Cruise Control shall turn on when the car has been turned on

1.2) Cruise Control shall begin logging activity when it receives power

1.3) System shall activate cruise control once activated by driver

1.4) System shall deactivate cruise control once deactivated by driver

1.5) System shall set speed to any setting given by the driver

1.6) System shall maintain the speed set by the user.

1.7) System shall increase speed of cruise control, given by speed increase input by driver

1.8) System shall decrease speed of cruise control, given by speed decrease input by driver

1.9) System shall activate within 0.5 seconds after activation from driver

1.10) System shall deactivate within 0.5 seconds after deactivation from driver or other sensors

1.11) System shall give confirmation that it has been engaged

1.12) System will confirm that all safety/functional requirements are met before the cruise control will be activated (these safety functions are checked with “safety” function, refer to UML)

1.13) System shall accept and use input from the user of the cruise control via input devices, such as desired knobs and buttons or information systems inside the car

1.14) System shall receive data continuously from sensors every 0.5 seconds of any brake application

1.15) System shall not activate if the speed is under 20MPH. (Information via pre-existing sensor in the car)

1.16) System shall accept all signals from any sensors on the car for approval to activate cruise control

1.17) System shall give approval request to sensors before activation

1.18) System shall have accuracy range from +1 and -1 mph of the set speed

1.19) System shall receive time and date from car clock every second

1.20) System shall return to target speed once the gas is applied and is finished accelerating.

1.21) System shall safely deactivate due to mechanical problems, such as malfunctioning parts as well as improperly functioning electronic electronics which its status as “working” will be verified by a separate function (in uml see “safety”) with the essential functions of the car or outside events that may warrant a change in speed

1.22) System shall stay on for exactly 2 seconds after the car shuts down

Non-Functional Requirements:

2.1) System shall activate and deactivate within 100ms of user input.

2.2) System software shall maintain a 4-9 reliability (99.99%)

2.3) System hardware shall maintain a 5-9 reliability (99.999%)

2.4) System shall not activate if any connections to the hardware fail.

2.5) System shall log all activities performed by the cruise control, all activities such as cruise control on, cruise control off, activation, deactivation, errors, and change in target speed all qualify as ‘events’. These ‘events’ will be put in the log along with the cruise control internal clock described in System Requirements s

System Requirements:

3.1) System will have the permissions and ability to access, modify, and manipulate the hardware and software that is required for normal function

3.2) Technician shall have the ability to access log files with physical interface

3.3) Technician shall be able to configure system once accessed

3.4) Cruise Control System shall draw power from the alternator and the battery to log events from initial activation until approximately 2 seconds after deactivation

3.5) System shall have an internal clock that will have the ability to access the time via interface with the car when the cruise control turns on and will keep its own timing until the car turns off. This internal clock will be for logging purposes so we can determine when errors arised or events occured.

3.6) System shall receive the current speed from EMS continuously every 1 second

3.7) System shall provide feedback that user has deactivated system

3.8) System shall provide feedback that user has set speed

3.9) System shall have activation request to throttle

3.10) System shall have deactivation request to throttle

3.11) System shall follow secure communication protocol given by car manufacturer, industry, and government

Security Requirements:

4.1) System shall not have access to the internet

4.2) System shall not have the ability to access systems outside of its scope of performance as denoted in the Functional Requirements sections

4.3) System software shall contain levels of security so that users can not alter aspects of the system such as the minimum speed.

4.4) System shall contain a user authentication system for retrieving the log

4.5) System shall not provide interface remote access

Use Case Definition

Use cases are a collection of user scenarios that describe the thread of usage of a system to help give others an idea of how it is meant to be used and how our cruise control will benefit them.

Use Case #1

- **Primary Actor :** The driver of the car
- **Goal:** Enable cruise control
- **Preconditions:** Seat belts are on, the car is on and moving, and the car has reached and maintained a speed that is at least as fast as the minimum speed required to use cruise control.
- **Scenario:**
    - Actor is driving and presses the cruise control button.
    - The cruise control system determines if the car is safe to enable cruise control (checking seat belts, speed, via sensors).
    - Once the car is deemed safe the cruise control system enables and awaits a set or change in speed.
    - Cruise Control System provides visual response for user to input speed
    - Actor inputs a desired set speed to the system.
    - Cruise control requests the EMS to set the speed to the set speed of the cruise control system
    - Time of Activation, and Speed set are logged to the system.

Use Case #2

- **Primary Actor :** the driver of the car
- **Goal:** Change the speed of the cruise control
- **Preconditions:** Cruise control is enabled and is working at a set speed. All safety requirements are also met.

- **Scenario:**
    - Actor is using cruise control and a set speed.
    - Actor increases or decreases the set speed via input to the system.
    - Cruise Control System provides visual response that the speed has been changed
    - Cruise control requests the EMS to set the speed to the set speed of the cruise control system
    - Time of speed changes, and the speed it was changed to is logged to the system.

<u>Use Case #3</u>

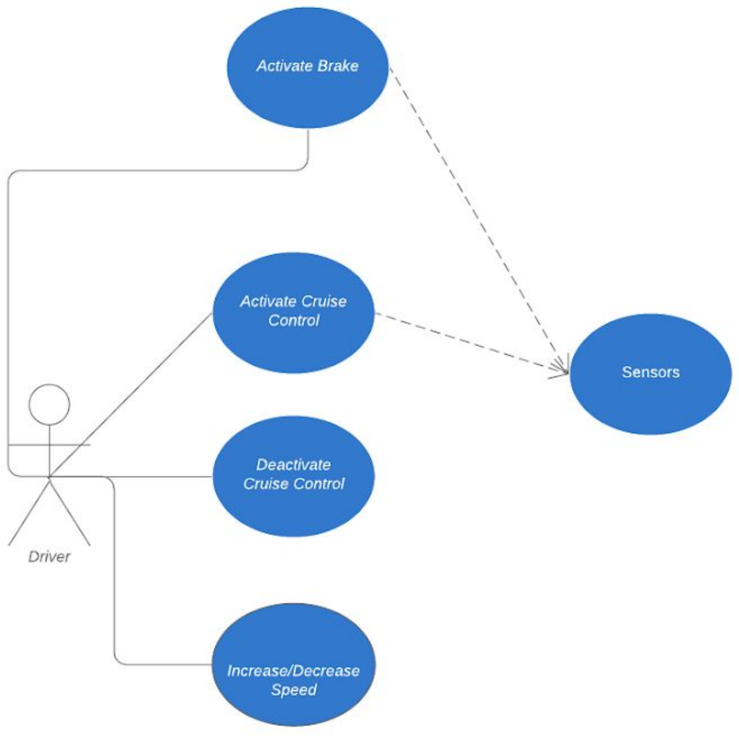
- **Primary Actor :** The driver of the car
- **Goal:** Manually disable cruise control
- **Preconditions:** Cruise control is enabled and is working at a set speed
- **Scenario:**
    - Actor is using cruise control and decides to disable it.
    - Actor either : applies the break, applies the gas, or presses disable cruise control button
    - Cruise Control System Requests the EMS to stop maintaining a speed
    - Cruise Control System provides visual response that Cruise control has been disabled.
    - Time of deactivation , and method of deactivation are logged to the system.

<u>Use Case #4</u>

- **Primary Actor :** The driver of the car
- **Secondary Actor :** Other drivers in other cars
- **Goal:** Automatically disable cruise control
- **Preconditions:** Cruise control is enabled and is working at a set speed

- **Scenario:**
  - Primary Actor is using cruise control and collides with a secondary actor on the road.
  - Sensors in the car detect that a collision has occurred, and sends that information to the cruise control system.
  - Cruise Control System Requests the EMS to stop maintaining a speed
  - Cruise Control System provides visual response that Cruise control has been disabled.
  - Time of deactivation , and method of deactivation are logged into the system.

Use Case Diagram



CRC Cards

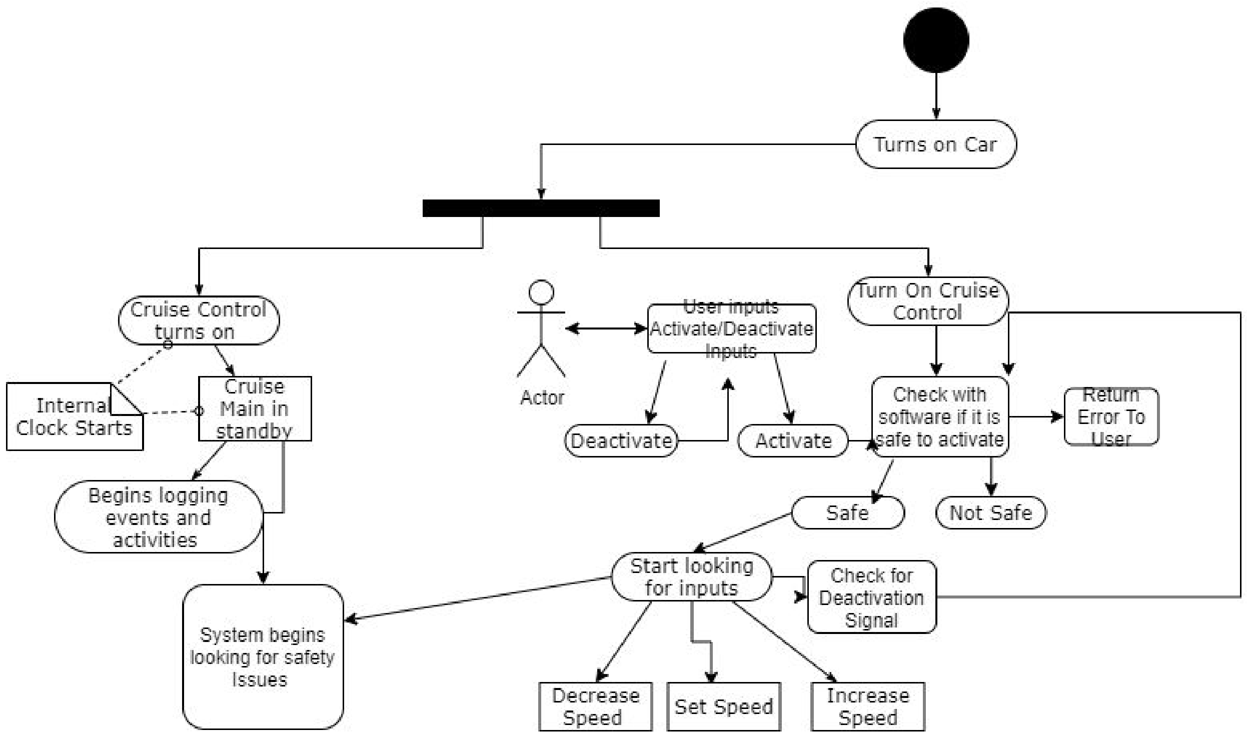
Car	Cruise Control, Safety, Sensors
• Preforming Correctly as a Car	• Sensors gather information about the status of the car

Safety	Sensors
• Checks with sensors such as: tire pressure, seat belts, weather, accelerator, and speedometer to gather information and tell the system if the cruise control has the ability to activate given the current status of all sensors • Constantly running during activation of the cruise control to ensure it does not preform in substandard conditions	

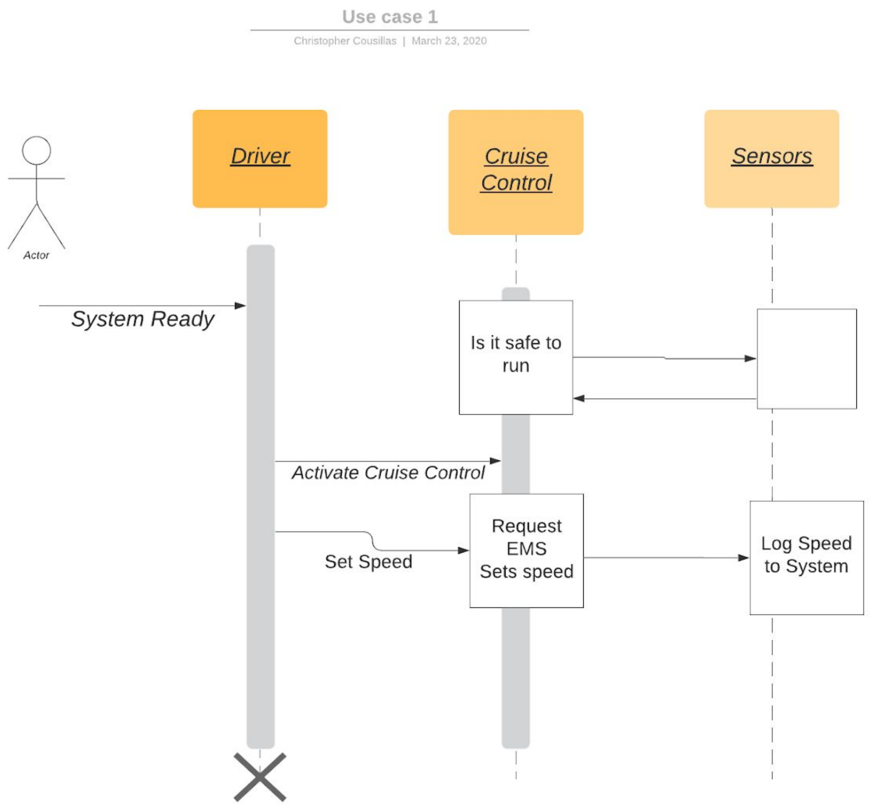
Cruise Control	N/A Sensors, Safety
• Accepting inputs such as Activate/Deactivate , increase speed, set speed, decrease speed • displaying and returning useful information to the user • Logging all activities that the user preforms • maintaining a system clock to time stamp logs • Accepting information from other in car systems and sensors	• Interacting with Sensors

Sensors	N/A
• Sensors such as TPMS, seat belt, temperature, speed, ect. are all tracked in this class • It is the responsibilities of the sensors to return correct and useful information in a timely manne	

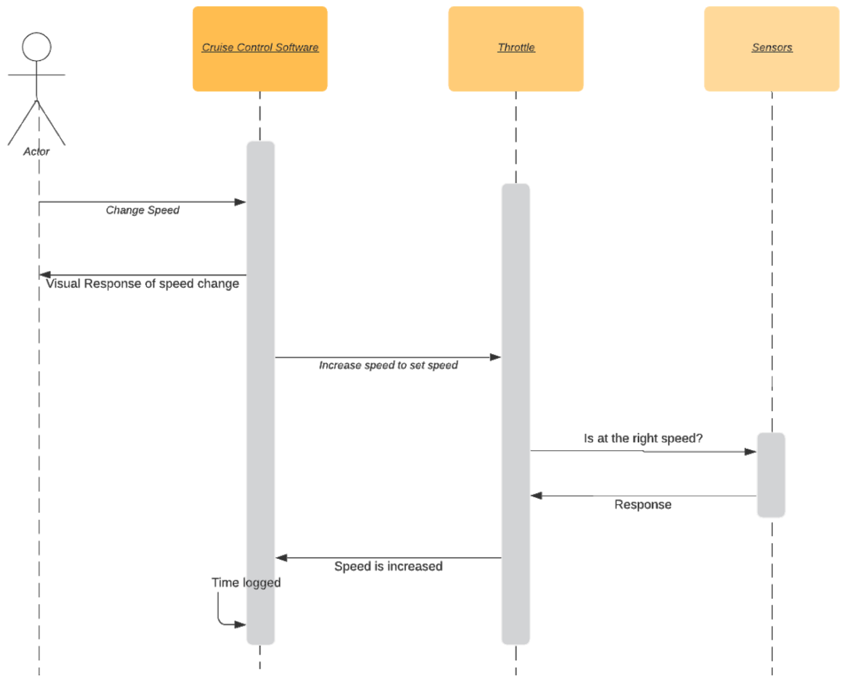
Activity Diagram



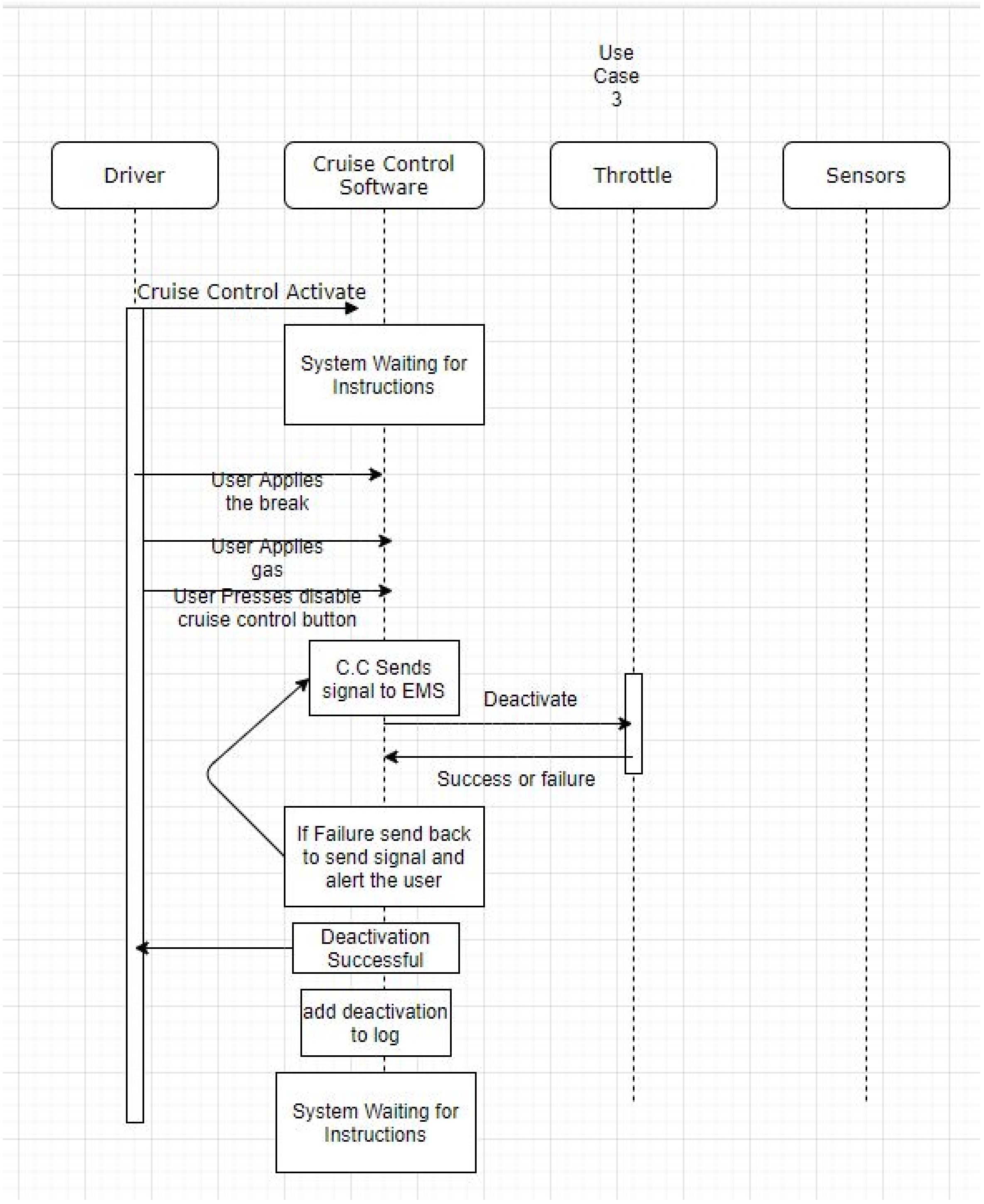
Sequence Diagrams

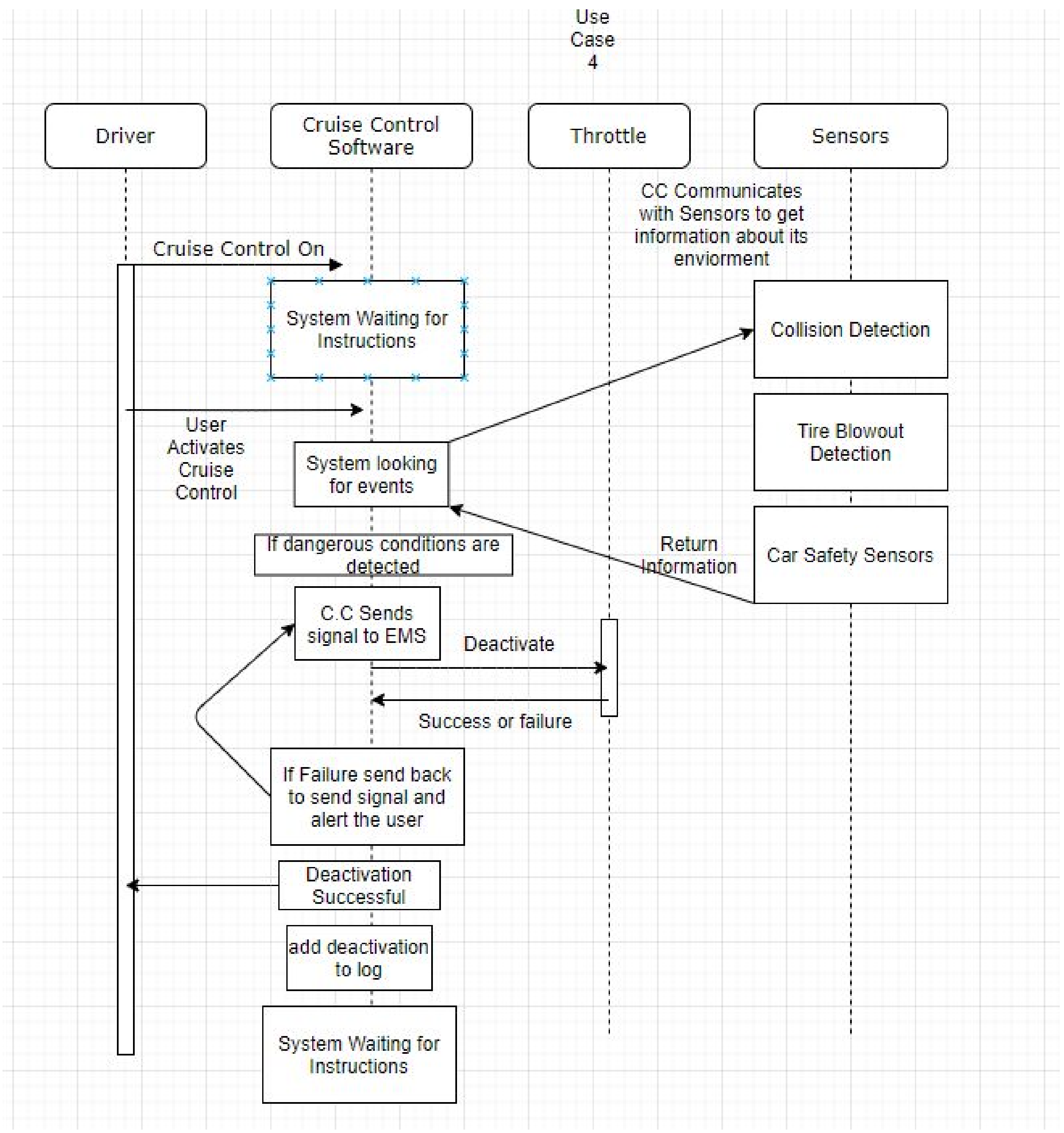


Use Case 2

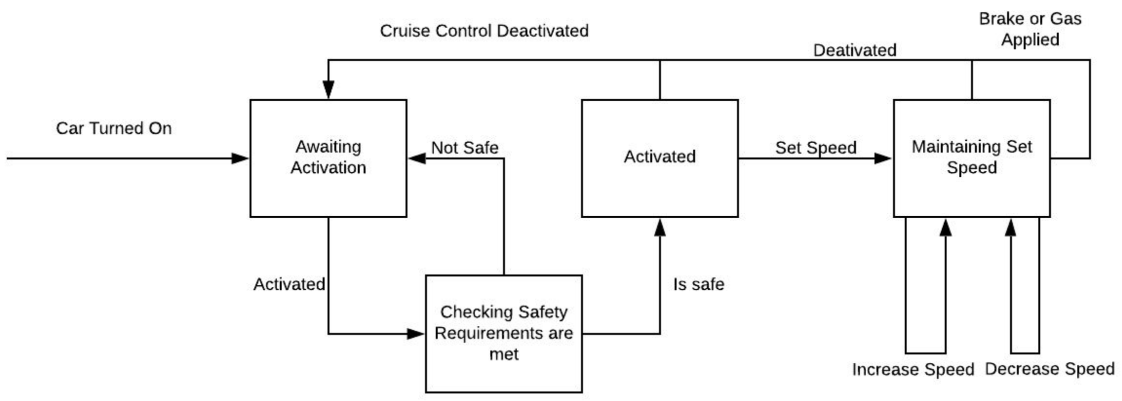


Cruise Control Software
Throttle
Sensors
Actor
Change Speed
Visual Response of speed change
Increase speed to set speed
Is at the right speed?
Response
Speed is increased
Time logged

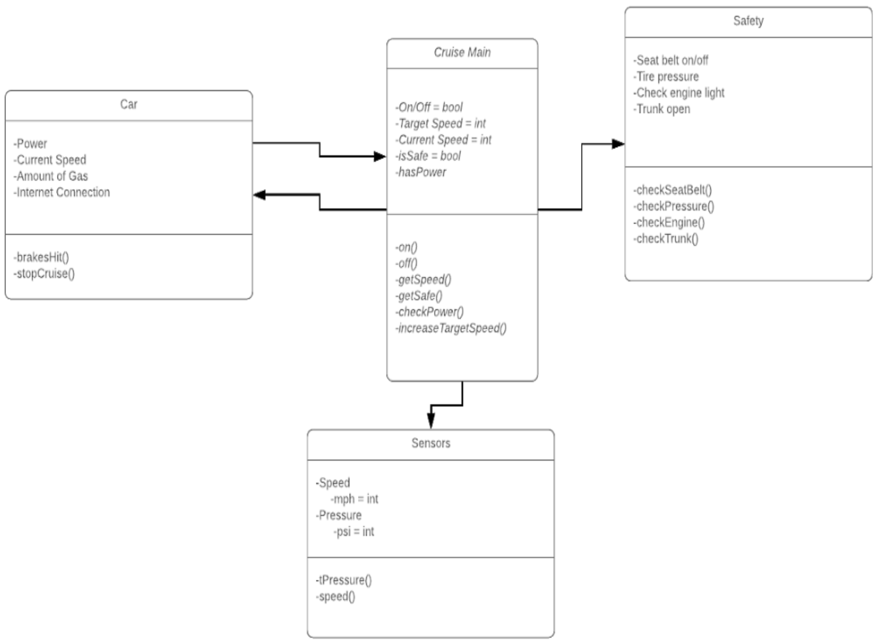

Use Case 3
Driver
Cruise Control Software
Throttle
Sensors
Cruise Control Activate
System Waiting for Instructions
User Applies the break
User Applies gas
User Presses disable cruise control button
C.C Sends signal to EMS
Deactivate
Success or failure
If Failure send back to send signal and alert the user
Deactivation Successful
add deactivation to log
System Waiting for Instructions


Use Case 4
Driver
Cruise Control Software
Throttle
Sensors
CC Communicates with Sensors to get information about its enviorment
Cruise Control On
System Waiting for Instructions
Collision Detection
User Activates Cruise Control
System looking for events
Tire Blowout Detection
If dangerous conditions are detected
Return Information
Car Safety Sensors
C.C Sends signal to EMS
Deactivate
Success or failure
If Failure send back to send signal and alert the user
Deactivation Successful
add deactivation to log
System Waiting for Instructions

State Diagram



UML State Diagram



<u>Section 5: Software Architecture</u>

<u>Section 1:</u>

1. Object Oriented
    a. Pros
        i. Code is easy to maintain
        ii. Modular, easy to add more features.
        iii. Easy to test each object
        iv. Simultaneous development is easy to implement
        v. Modularity also promotes reusability
        vi. Easy to maintain code because of its structure
    b. Cons
        i. Code can become bloated and unnecessarily extensive for a simple system, becoming inefficient
        ii. As a result of bloated software and increased complexity, this can prove to be computationally inefficient
        iii. Because of the reusability, this can also lead to duplication of like methods and objects
2. Layered Architecture
    a. Pros
        i. Code is easily testable, you can test each layer individually.
        ii. Clean data flow : data only moves up or down the layers.
        iii. Ideas and tasks are separated by layer
    b. Cons
        i. Performance can suffer as more layers are added
        ii. Complexity increases with the number of layers added
        iii. Abstraction can alter the desired course of the layer
        iv. All the objects of the same kind are grouped together which makes them easy to find

        v. Consistency

3. Model View Controller Architecture
    a. Pros
        i. Describes explicitly how the Client interacts with the system
        ii. Describes uniquely how data can influence and interact with the system
        iii. Easy to update separate parts of the system
        iv. Promotes collaboration between developers because of its explicitly defined interaction between the different pieces of software
    b. Cons
        i. This structure is very complicated
        ii. Requires many updates

Section 2:

Our architecture will be the object oriented architecture. Our structure will contain several objects and subclasses that will interact with one another to properly communicate and properly function according to our requirements document. Our object will interact with each other via method calls and a simple and easy to follow hierarchy. Most of our classes will interact and be connected to our bridge object.
This will allow interaction and collaboration with other objects in our architecture.

Our object oriented architecture will additionally be the most advantageous for us with its modularity, that promotes reusability and modularity, as well as its ability to be synergistic with parallel development. Although it is known that this structure/hierarchy may lead to an increase in complexity, we believe that we can stick to and maintain our structure. By doing this we will mitigate this risk and prevent its consequences.

Section 3:

- How is control managed within the architecture?

  The control begins in the main program (the cruise control executive). It then flows to each of the subprograms. Each component of the architecture accesses a bridge in order to communicate with the subprograms. This essentially centralizes the control to the bridge.

- Does a distinct control hierarchy exist, and if so, what is the role of components within this control hierarchy?

  Yes, a distinct control hierarchy exists. It starts at the main program and continues to the subprograms. At each step the control must be transferred through the bridge.

- How do components transfer control within the system?

  Control is not transferred as it is shared through the bridge. Each component can send data to and from the bridge at any point.

- How is control shared among components?

  Control is shared via a bridge object that can call other components. The bridge is essentially a connection from each to component that also holds all concurrent data on the systems state. Each component at any point can access other components through this bridge, thus maintaining control.

- What is the control topology (that is the geometric form that the control takes)?

  The control topology is spherical as it centers around the bridge. The ever component can have direct control through the bridge, making the bridge the control center.

- Is control synchronized, or do components operate asynchronously?

Components operate asynchronously as functions must work off of one another in the system. As data is read from the EMS, it is sent to the bridge, where other components can access and execute changes to the system through the bridge.

Section 4:

Our data architecture is composed of models, policies, rules, or standards that govern which data is collected, and how it is stored, arranged, integrated, and put to use in data systems and in organizations.

•How are data communicated between components?

In order to communicate between components, a bridge will be utilized that will maintain variables and data on the current state of the system. In order to communicate to another component, it must do so through the bridge.

•Is the flow of data continuous, or are data objects passed to the system sporadically

The flow of data is not continuous, when the user provides input, or the system receives outside input, the data is sent to the bridge, so that each component can access it. Data is only passed when a change to the system is made.

•What is the mode of data transfer?

Data is transferred through a method call on the bridge.

•Do data components exist, and if so, what is their role?

Data components exist in the system for the transfer of data from other database programs. There will be a class that enables the systems components to access other components through a bridge.

•How do functional components interact with data components?

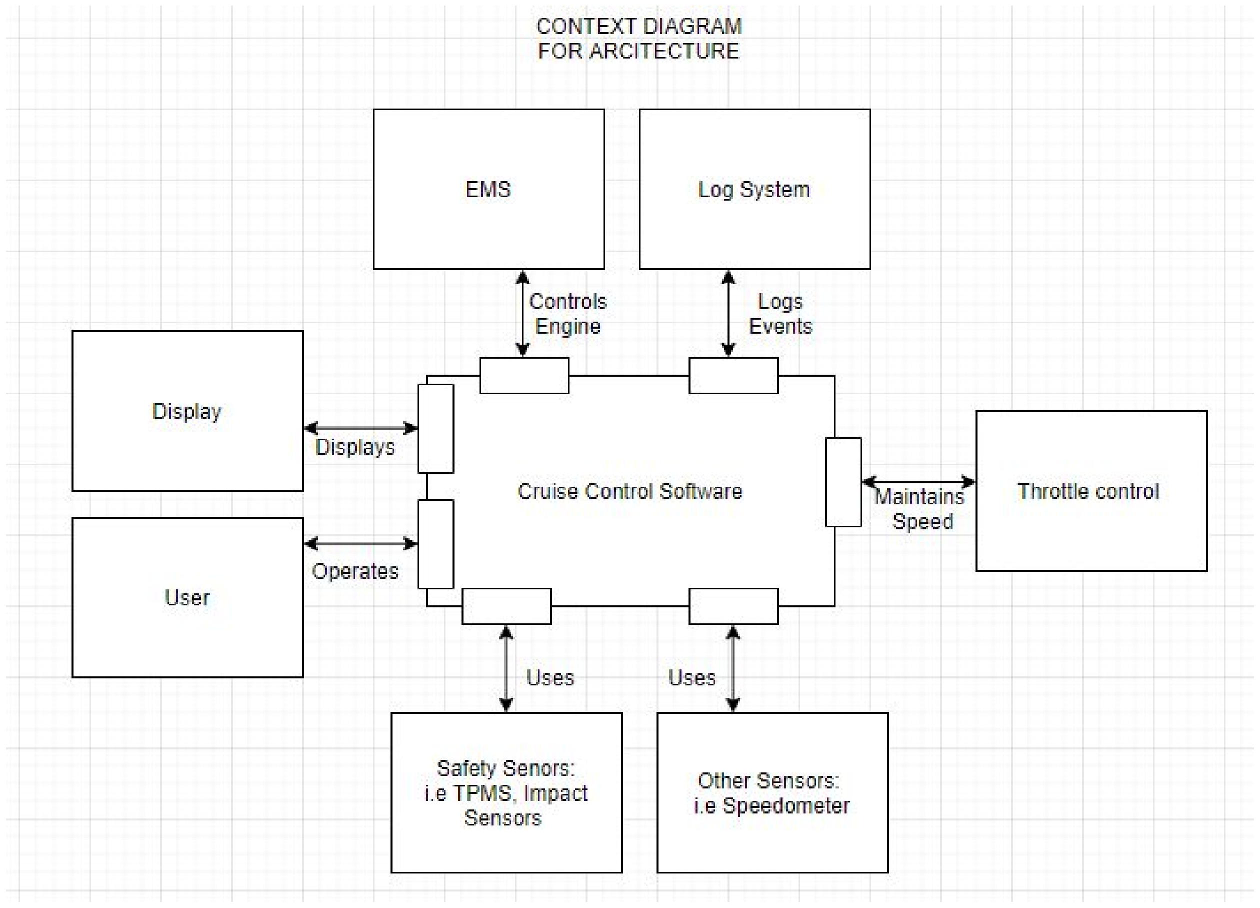
In this system, The functional components can access the bridge in order to have access to the system's data. The bride will be a data component and will also act as a connector to other functional components.

•Are data components passive or active?

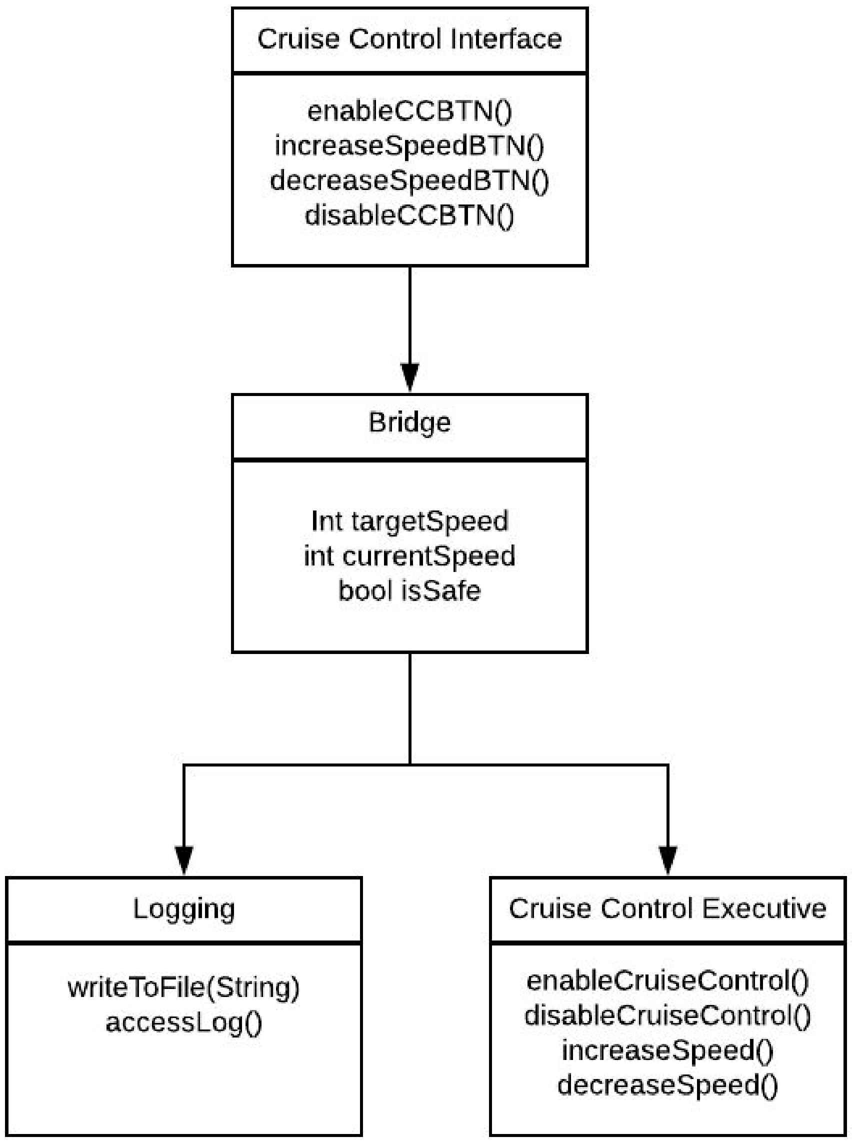
Data components in this system are active, it is updated every time something about the car changes.

<u>**Section 5:**</u>

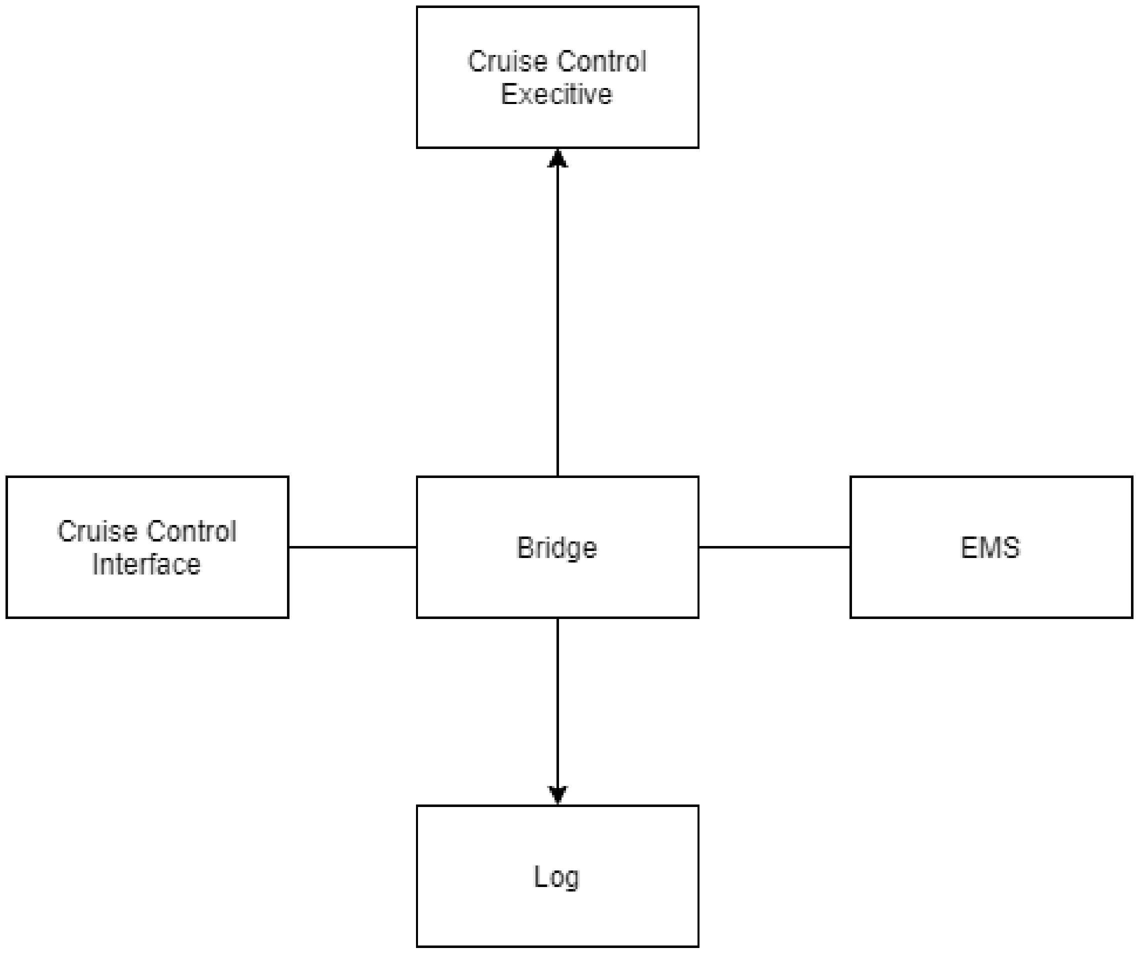
Architectural Designs:



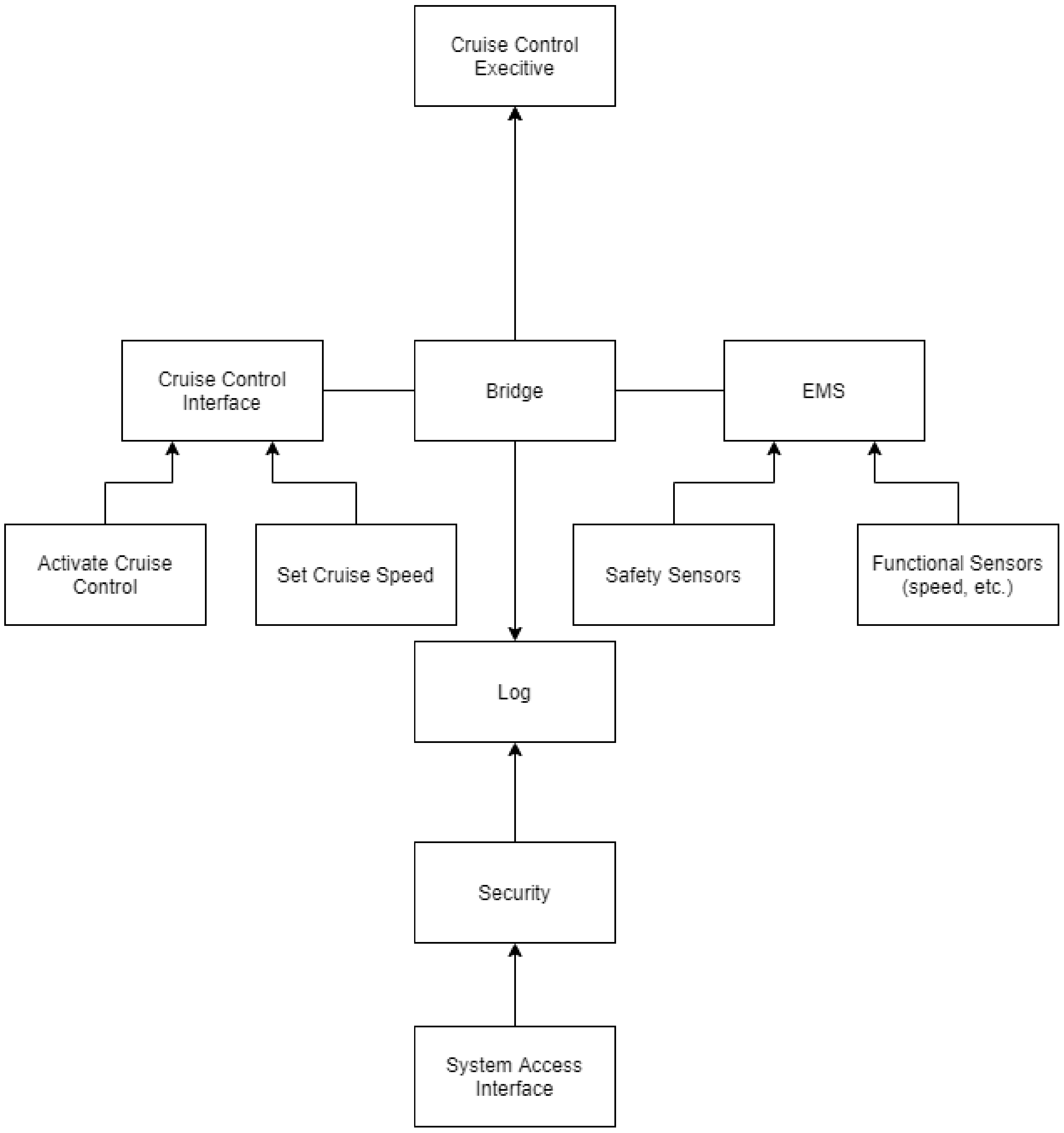
Function Archetype



Top Level Component Architecture:



Refined Component Architecture:



Section 6:

Short summary of issues discovered:

In the requirements section, we lack a place to generate logs and store them. In our architecture this will be done in the core layer, which will have access and have the ability to gather information about events that occur throughout the system. It will be implemented in a way for the system to record any events that occur in the system. This system will also date and time all of the records that it produces in order for it to be used in diagnostics.

Currently any clock logging information isn't implemented or expanded upon in the archetype section of the document. Information that is needed to be collected from and to the clock will be handled. We will create a clock that handles all variables and displays the information for the user.

Another fault of the current architecture that we found is that there is currently no security layer to prevent unauthorized access to system variables.We will come up with a functional security layer that will deal with authorizing a user reading or sending data through the layers of our architecture.

As stated in our requirements, a technician shall have the ability to access log files with physical interface. However, currently we lack a method of giving data to the technician. A way for a technician to access the core layer securely and access the data, will be implemented.

Section 6 Code:

cc_bridge.java:

```
public class cc_bridge { // set global variables to the cc_bridge class
	static boolean targetSet = false;
	static boolean isEnabled = false;
	static int targetSpeed;
	static int currentSpeed = 0;
	static boolean isSafe = true;



	//function to set the target speed of the CC
	public static void setTargetSpeed(int speed) {
		targetSpeed = speed;
		cc_log.writeToFile("Cruise Control : Target speed set to " + targetSpeed);
		cc_executive.onSpeedChanged();
	}
	//set to current speed
	public static void setCurrentSpeed(int speed) {
		currentSpeed = speed;
		cc_log.writeToFile("Cruise Control : Current speed is " + currentSpeed);
		cc_executive.onSpeedChanged();
	}
	//checks the global variables
	public static void setIsEnabled(boolean enabled) {
		isEnabled = enabled;
		if(!isEnabled) {
			targetSet = false;
			}
			cc_log.writeToFile("Cruise Control System has been " + (isEnabled ?
"enabled" : "disabled"));
		}

}
```

cc_executive.java:

```
public class cc_executive {

//enable cruise control checking the minimum speed has been met
	public static void enableCruiseControl() {
		if(!cc_bridge.isEnabled && cc_bridge.currentSpeed >= 20) {
			cc_bridge.setIsEnabled(true);
		}
	}
	// Disables the Cruise Control
	public static void disableCruiseControl(){
		if(cc_bridge.isEnabled) {
			cc_bridge.setIsEnabled(false);
		}
	}
	//increase the set speed, so long as it maintains the speed requirements
	public static void increaseSpeed() {
		if(cc_bridge.isEnabled && cc_bridge.targetSpeed < 200) {
			if(!cc_bridge.targetSet) {
				cc_bridge.targetSet = true;
				cc_bridge.setTargetSpeed(cc_bridge.currentSpeed+1);
				}else {
					cc_bridge.setTargetSpeed(cc_bridge.targetSpeed+1);
				}
			}
		}

//decrease the set speed, so long as the speed requirements are still met
		public static void decreaseSpeed() {
			if(cc_bridge.isEnabled && cc_bridge.targetSpeed -1 >= 20) {
				if(!cc_bridge.targetSet) {
					cc_bridge.targetSet = true;
					cc_bridge.setTargetSpeed(cc_bridge.currentSpeed-1);
				}else {
					cc_bridge.setTargetSpeed(cc_bridge.targetSpeed-1);
				}


			}
		}
		//logic to handle the current speed changing, and setting it back to the set speed
		public static void onSpeedChanged() {
			if(cc_bridge.currentSpeed > cc_bridge.targetSpeed &&
cc_bridge.targetSet) {
				cc_log.writeToFile("Cruise Control : Decelerating to target speed");
				cc_bridge.setCurrentSpeed(cc_bridge.targetSpeed);
			}else if(cc_bridge.currentSpeed < cc_bridge.targetSpeed &&
cc_bridge.targetSet) {
				cc_log.writeToFile("Cruise Control : Accelerating to target speed");
				cc_bridge.setCurrentSpeed(cc_bridge.targetSpeed);
			}
```

```
        }

}
```

cc_interface.java

```
//import all the necessary resources to produce a GUI
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class cc_interface extends JPanel {
//global variables
   private JButton enable;
   private JButton disable;
   private JButton brake;
   private JButton gas;
   private JButton jcomp5;
   private JButton jcomp6;
   private JButton accessLog;
   private static JTextArea log;

   public cc_interface() {
      //construct components
      enable = new JButton ("Enable Cruise Control");
      disable = new JButton ("Disable Cruise Control");
```

```
brake = new JButton ("Brake");
gas = new JButton ("Gas");
jcomp5 = new JButton ("Increase Target Speed");
jcomp6 = new JButton ("Decrease Target Speed");
accessLog = new JButton("Access Log");
log = new JTextArea (5, 5);

//adjust size and set layout
setPreferredSize (new Dimension (944, 601));
setLayout (null);

//add components
add (enable);
add (disable);
add (brake);
add (gas);
add (jcomp5);
add (jcomp6);
add(accessLog);
add (log);

//set component bounds (only needed by Absolute Positioning)
enable.setBounds (15, 20, 235, 65);
disable.setBounds (15, 100, 235, 65);
brake.setBounds (15, 210, 155, 365);
gas.setBounds (265, 210, 165, 365);
jcomp5.setBounds (260, 20, 170, 65);
jcomp6.setBounds (260, 100, 170, 65);
accessLog.setBounds(450, 20, 485, 65);
```

```
        log.setBounds (450, 100, 485, 470);

//button listeners to add functionality to GUI buttons
        enable.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                cc_executive.enableCruiseControl();
            }
        });

        disable.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                cc_executive.disableCruiseControl();
            }
        });

        jcomp5.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                cc_executive.increaseSpeed();
            }
        });

        jcomp6.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                cc_executive.decreaseSpeed();
            }
```

```
    });

    brake.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            cc_executive.disableCruiseControl();
            if(cc_bridge.currentSpeed - 5 >= 0) {
                cc_bridge.setCurrentSpeed(cc_bridge.currentSpeed - 5);
            }
        }
    });

    gas.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            cc_bridge.setCurrentSpeed(cc_bridge.currentSpeed + 5);
        }
    });

    accessLog.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
                String pass= JOptionPane.showInputDialog("Enter Password: ");
                cc_log.accessLog(pass);
        }
    });

}
```

```
//for the case that an incorrect password is inputted
public static void badPassword() {
      JOptionPane.showMessageDialog(null, "The password is incorrect" );
   }

//append the log to the gui
public static void setLog(String str) {
        log.append(str + "\n");
   }

   public static void main (String[] args) {
        cc_log.initLog();
      JFrame frame = new JFrame ("Cruise Control");
      frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
      frame.getContentPane().add (new cc_interface());
      frame.pack();
      frame.setVisible (true);
      cc_log.writeToFile("Car Turned On : Awaiting Input");
   }
}
```

cc_log.java

```
import java.awt.Desktop;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```
public class cc_log {

//open log, if not present, create the file
  public static void initLog() {
    try {
       File myObj = new File("ccLog.txt");
       if (myObj.createNewFile()) {
        System.out.println("File created: " + myObj.getName());
       } else {
        System.out.println("File already exists.");
       }
      } catch (IOException e) {
       System.out.println("An error occurred.");
       e.printStackTrace();
      }
  }


  public static void writeToFile(String str) {
//generate new format for date and time, produce new object
    SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    Date date = new Date();
    String output = format.format(date);
    str = output + " " + str;
    System.out.println(str);
    cc_interface.setLog(str);
     //add to the file
    appendStrToFile("ccLog.txt", str);
  }
```

```
public static void appendStrToFile(String fileName, String str) {
            try {

            // Open given file in append mode.
                    BufferedWriter out = new BufferedWriter(
                    new FileWriter(fileName, true));
                    out.write(str + "\n");
                    out.close();
            }
            catch (IOException e) {
                    System.out.println("exception occurred" + e);
            }
    }


public static void accessLog(String password) {
   //ask for password, pass it into this method
   if(password.equalsIgnoreCase("password")){
      try {
                            openLog();
                    } catch (IOException e) {
                            e.printStackTrace();
                    }
   }else {
      cc_interface.badPassword();
   }
}
```

```
    public static void openLog() throws IOException {
        // open the log for the user

        File file = new File("ccLog.txt");

        //first check if Desktop is supported by Platform or not
        if(!Desktop.isDesktopSupported()){
            System.out.println("Desktop is not supported");
            return;
        }

        Desktop desktop = Desktop.getDesktop();
        if(file.exists()) desktop.open(file);

    }
}
```

Section 7:

Test Case #1 : Increase speed to 50, enable cruise control, increase set speed, disable cruise control.

Result: Successful

Test Case #2 : Brake and Gas multiple times to test mechanics

Result: Successful

Test Case #3: Test cruise control cannot be activated under 20MPH

Result: Successful

Test Case #4 : Test cruise control cannot be brought under 20MPH after activation

Result: Successful

Test Case #5 : Test brake to disable cruise control

Result: Successful

Test Case #6 : Activate log with password, “password”, system shall authenticate and open the log.

Result: Successful

Test Case #7: Activate log with and other password, system shall not open the log

Result: Successful

Test Case #8: Bring Speed up to 20MPH and activate cruise control. Increase set speed to 30, cruise control shall begin maintaining a speed of 30MPH

Result: Successful

Test Case #9: Cruise Control displays results of events occurring via the GUI

Result: Successful

Test Case #10: Bring Speed up to 30MPH and activate cruise control, decrease the set speed to 25MPH, the car shall begin maintaining a speed of 25MPH.

Result : Successful

Test Case #11: Note the date and time of the log, and compare them to the system date and time to check accuracy

Result : Successful

Test Case #12: If no such file exists, the cruise control log function properly generates a log file with proper contents.

Result: Successful

Test case #13 : Bring speed up to 25MPH and activate cruise control, apply the gas to increase speed to 30MPH, the cruise control system shall decelerate back to 25MPH and maintain that speed.

Result : Successful

Test case #14: All buttons successfully call events

Result: Successful

Test case #15 : Start user software, GUI should present activation notification.

Result : Successful

Section 8:

Document Issues:

1. Executive Summary - wording needs to be changed so it is more concise
2. Introduction - consistency issues
3. Reliability - specification of hardware reliability needs to be removed because it is outside the scope of our development
4. Issues that were discovered in section 5 of the Software Architecture Section of document
5. Software Architecture - Issues:
   a. Section 6 of the Software Architecture needs to be consolidated
   b. Functions, classes, data types, and variables need to be specified
   c. Architectures need to be adjusted based on the above changes

Document Resolutions:

1. Executive Summary - changed wording to be more concise, fixed grammar
2. Introduction - fixed consistency issues with terms used.
3. Reliability - removed specification of hardware reliability as that is outside the scope of our development.
4. Team members implemented fixes to issues discovered in section 5 of the Software Architecture
5. Fixed issues that existed within our Software Architecture portion of our Cruise Control Prototype Document
   a. Consolidating section 6 of the Software Architecture
   b. Specify functions, classes, data types, and variables
   c. Adjust architectures based on the above changes

Code Issues:

6. System was able to “brake” into negative numbers
7. Cruise Control Log was only showing the most recent logged message
8. System was able to cruise control below 20mph after starting above 20 (by lowering the set speed)
9. Date and time won't format correctly
10. File will not print the log information
11. Log file failed to initialize

Code Resolutions:

6. Added system check to maintain minimum and maximum car speed
7. Switched to a different file writing API
8. Added system check to maintain minimum and maximum cruise control speed
9. Used a special “SimpleDateFormat” and import java.util.date
10. Switched to a special file open in java
11. Created a method with error checking to initialize the log