Software Requirements Specification (SRS)

Pedestrian Collision Avoidance System (PCAS)

Authors: Caroline Gormely, Chris Belack, Kyle Ernster, Bram Kineman, Riley Thompson

Customer: Mr. Chris Capaldi

Instructor: Dr. Betty H.C. Cheng

1 Introduction

According to the Center for Disease Control, in 2017, 5,977 pedestrians were killed in traffic crashes in the United States [1]. One cause of pedestrian-vehicle collisions is drivers failing to yield the right of way to pedestrians. In addition, pedestrians may abruptly run into the road. As a result, the driver may not have ample time to stop the vehicle, causing a collision. In conclusion, new systems are needed to ensure vehicles do not collide with pedestrians. The Pedestrian Collision Avoidance System (PCAS) aims to make vehicles safer by ensuring there are zero pedestrian-vehicle collisions while cruise control is enabled. The PCAS recognizes and tracks pedestrians in the vehicle’s path and will cause the vehicle to automatically brake when it determines that a pedestrian-vehicle collision is imminent. This will mitigate the problem of drivers failing to yield the right of way to pedestrians as well as failing to maintain attention while operating the vehicle under cruise control, therefore reducing the number of pedestrian-vehicle collisions.

This document describes the requirements, functionality and different uses of the PCAS. Models and a prototype are also included to visually demonstrate how the PCAS will behave under different scenarios. Section 1 details the purpose of this document and the scope of the software used in the PCAS.

1.1 Purpose

The purpose of the Software Requirements Systems (SRS) is to provide clear descriptions of the PCAS and its intended behavior for the customer Dataspeed. The SRS is intended to ensure the customer, clients and development team are in agreement about the design, purpose and implementation of the PCAS. It is important for the parties listed above to have consistent expectations in order to build a system that satisfies the requirements: a system that ensures zero pedestrian-vehicle collisions while drivers operate with cruise control enabled.

1.2 Scope
The Pedestrian Collision Avoidance System is built with the intent of guaranteeing zero pedestrian-vehicle collisions. The PCAS is an embedded system in an automotive vehicle and therefore is constrained to the capabilities of the vehicle.

PCAS includes pedestrian recognition and tracking and collision mitigation features whenever the system is on. The driver is able to override the system, though, if desired. Whenever the system is on, the path in front of the vehicle is being continually monitored for pedestrians. When the sensor detects a pedestrian, the sensor sends the pedestrian data to the PCAS so it can determine if a pedestrian-vehicle collision is imminent. If a collision is imminent, the vehicle will begin to automatically decelerate. In addition, the speaker and Heads-up-Display (HUD) will audibly and visually notify the driver of such detection and deceleration.

1.3 Definition, acronyms, and abbreviations

- **Pedestrian Collision Avoidance System (PCAS):** The embedded system described throughout the document.
- **Pedestrian-Detection-Sensor (PDS):** Stereo camera that detects pedestrians and sends pedestrian location data to the PCAS.
- **Brake-by-Wire (BBW):** Automatically decelerates the vehicle when a potential pedestrian-vehicle collision is detected.
- **Heads-up-Display (HUD):** Visually notifies the driver of system failure and automatic deceleration.
- **Software Requirements Specification (SRS):** This document outlining the requirements of the PCAS as interpreted by the team.

1.4 Organization

The remainder of this document is organized as follows. Section 1 describes the system and its scope. Section 1 also describes the purpose, definitions, acronyms, abbreviations and organization of the paper.

Section 2 describes the PCAS at a high level. Product perspective, product functions, user characteristics and constraints, assumptions and dependencies are all included in this section.

Section 3 consists of the requirements of the system. They are organized hierarchically and listed in order of importance.

Section 4 contains all of the models and diagrams. They consist of a Use Case diagram, Domain Model, Sequence Diagrams and State Diagrams.

Section 5 describes the prototype and explains how to run it. Some scenarios included in the prototype are then described.
Section 6 contains the references for this document.

2 Overall Description

The following section outlines the key aspects of the PCAS and how the embedded system will work at a high level. Section 2.1 focuses on the scope of the product and a high level view of the entire system. Section 2.2 expands on each of the key components in the embedded system and a more low level approach. Section 2.3 depicts the typical driver for the system as well as their expected experience level and section 2.4 discuss potential constraints of the system. Section 2.5 discusses the assumptions and dependencies that can be made in the development of this system while section 2.6 provides a glimpse into what future patches may be able to improve on.

2.1 Product Perspective

The PCAS is meant to add an additional layer of autonomy to vehicles with cruise control by providing a feature that is able to detect and mitigate imminent collisions between the vehicle and a pedestrian without driver involvement. Such a system should increase safety and greatly reduce the number of pedestrians killed in traffic annually.

The scope of this project focuses on an embedded system within a vehicle and more specifically on the development of the Pedestrian Collision Avoidance (PCA) algorithm. The PCA will communicate with other elements of the embedded system, but will only use them as input and output operations. The input will be used in determining if a collision is imminent and the output will be used to facilitate braking requests. The following figure depicts the PCA algorithm and furthermore how it fits into the entire system domain.

![System Architecture](image)

Figure 1: Pictorial Representation of the PCAS
There are many ways in which a driver will interact with the system. Specifically, once a driver decides to turn on cruise control, our system will also turn on. Turning cruise control off will inherently also turn the PCAS off. However, features are in place for a driver to intuitively take back control of the vehicle without having to turn off cruise control. If a driver pulls the brake lock or presses either pedal, the PCAS will shut off. Consequently, it will turn back on if the brake lock or either pedal is released and cruise control is enabled. These features allow the driver the ability to stop on their own. It is when they do not interfere that the system will take over and avoid the collision.

In regards to hardware constraints, as illustrated in Figure 1, each vehicle will need a pedestrian detection sensor, which is a standard stereo camera. Additionally, each vehicle will need to possess a Brake-by-Wire system. An interface for a driver notifier must also be present for the vehicle's dashboard. Lastly, the ability for the vehicle to have a safety controller will be required.

With respect to software constraints, each vehicle will need to possess software which allows the embedded system to communicate. Furthermore, every element within Figure 1 will need such abilities. The pedestrian detection sensor will need to be able to send information to the PCA algorithm and the PCA algorithm will need to be able to send requests to the Brake-by-Wire system. Finally, every vehicle will need the ability to communicate it’s current speed with the PCA algorithm.

Ultimately, each component of the PCAS will need to be limited in bandwidth to a reasonable rate for an automobile. Similarly, the size of a vehicle’s memory will be a limiting constraint in the development of the PCAS.

2.2 Product Functions

The PCAS aims to provide enhancements in safety for both the driver of the vehicle as well as surrounding pedestrians by both recognizing potential collisions and mitigating them. This system's main objective is to ensure driver and pedestrian safety as well as minimize time lost in doing so. Namely, there are two key features in successfully developing the PCA algorithm: efficiency and effectiveness. In regards to effectiveness, the algorithm is effective if and only if there are zero collisions allowed under a specific certain ten scenarios. One system should be used to measure performance against all ten scenarios. Such scenarios are defined in a prototype in following sections. In regards to efficiency, the algorithm is efficient if and only if lost time is minimized for all non-collision scenarios. As a definition, lost time for this system is defined as the time difference in seconds between the system on and off to reach a common point beyond the pedestrian controlled vehicle back again at steady state velocity.
It is worth noting that the PCA algorithm must be constructed to assume it does not know which scenario is occurring. In other words, the only available pedestrian information will be coming from the pedestrian detection sensor. It is also important to note optimization should be a consideration in the development of this system. It should be able to compete with other systems of similar variation.

In order to achieve this, there are several main components of the embedded system that come into play. The pedestrian detection sensor is a stereo camera which outputs pedestrian recognition and tracking. These outputs will be of the form of pedestrian location as an x and a y coordinate relative to the vehicle with accuracy of $+/-0.5m$ as well as pedestrian velocity, which will consist of speed and direction. Speed will be with accuracy of $+/-0.2\frac{m}{s}$ and direction will be of accuracy $+/-5\ degrees$. These above signals will be sent to the PCA algorithm as a packet every 100 $ms$. Additionally, the vehicle speed will be provided to the PCA algorithm. The PCA algorithm will work in congruence with the safety controller to determine if there is an imminent collision and then administer brake requests to the Brake-by-Wire in order to safely stop the vehicle.

The Brake-by-Wire is a subsystem which is capable of responding to brake requests by interrupting the steady state velocity control, specifically cruise control, and then applying brake torque via electro-mechanical actuators at all four wheels of the vehicle, and sensing the actual vehicle deceleration for closed loop control. The Brake-by-Wire subsystem has deceleration accuracy of $+/-2\%$ as well as a response time to reach the decelerated request of 200 $ms$. It is also worth noting that a fail operational mode for the braking system is present and increases the response time to reach the decelerated request from 200 $ms$ to 900 $ms$. This feature is intended as a fail-safe which can be used to thwart security risks as well as mitigate any computational or sensor errors that may cause the system to collide with a pedestrian. Lastly, the Brake-by-Wire system has a release time of 100 $ms$ as well as a maximum deceleration of $0.7\ g$ where $g=9.81\ \frac{m}{s^2}$, the standard notation for the constant gravity.

Another key component of the embedded system is the vehicle itself. Across all scenarios, the vehicle will have a normal steady state speed of 50 $kph$ or $13.9\ \frac{m}{s}$. It will also have constant width, which can be thought of as the collision zone, which is equal to 2 $m$. In regards to efficiency and lost time minimization, the vehicle will have an acceleration back to steady state speed after a brake request of $0.25\ g$.

If a collision is imminent, in addition to a brake request resulting in the stoppage of the vehicle and the avoidance of a collision, the system incorporates an alert feature to notify the driver that it is taking over. There is a heads up display which will play an alert message over the speaker as well as display a warning light on the dash. It is important to note that the driver will also be
given intuitive ways of deactivating the PCAS and taking back control of the vehicle such as pressing a pedal or pulling the lock brake.

2.3 User Characteristics

A typical driver is expected to be licensed to operate a vehicle and have the ability to turn on cruise control. Additionally, a driver should understand that the PCAS turns on during cruise control and plays an alert message when avoiding a collision with a pedestrian. Such characteristics will be displayed in an owner’s manual, allowing virtually anyone who can operate a vehicle the ability to use the PCAS.

2.4 Constraints

There are a few constraints to this system that are noteworthy. First, the PCAS is only active if and only if cruise control is enabled. Since the development of the PCAS assumes that the pedestrian will always be detected by the sensor, the system should be considered only effective under great weather conditions. Otherwise, the driver will be expected to clear the pedestrian detection sensor. If the sensor is covered, the system will alert the driver and shut off until it is cleared. The driver should not assume the system will continue to work if a notification is on.

2.5 Assumptions and Dependencies

For the implementation of the PCAS to be successful, there are several aspects of the vehicle and driver that must hold true. From a software standpoint, all components of the embedded system must be effective and reliable in their communication. It is assumed that the vehicle will be able to send signals to the PCA algorithm in order to update its current speed. The pedestrian detection sensor must be efficient and correct in its signals to the PCA algorithm as well. Additionally, the Brake-by-Wire system must be able to properly handle brake requests. The PCAS also depends on a properly installed and operative cruise control. In addition to these subsystems working from a software standpoint, they all must also be working from a hardware standpoint. All components must be properly installed and tested for use.

Finally, it is expected that the driver is able to properly use the PCAS as well as cruise control. A user guide will be provided to those who purchase vehicles with the PCAS, but it is important to note that the proper use of the PCAS will heavily depend on drivers reading the manual as knowledge of the system is important for use.

2.6 Apportioning of Requirements
Future developments should look to increase upon the pedestrian detection sensor to account for poor weather conditions. Better sensors could be developed to the point where the driver does not need to clear the camera and thus the system could add more autonomy. Lastly, the addition to the PCA algorithm of accounting for pedestrians moving at non-right angles could be considered, as this system only considers scenarios where the pedestrian is moving at an exact right angle to the vehicle.

3 Specific Requirements

The following section contains a hierarchical list of requirements for the PCAS. There are requirements of software and hardware that already exist on the vehicle but will interact with PCAS. These requirements must be incorporated into the existing components for PCAS to behave properly. The remaining subsections include a list of global invariant requirements and primary requirements broken up by software, hardware, and security.

1. **Global Invariant Requirements**

   1.1. There must be zero vehicle-pedestrian collisions under the scenarios outlined in this document.
   1.2. The system will not require any human intervention to avoid vehicle-pedestrian collisions.
   1.3. The system will prioritize zero collisions, while minimizing lost time. The system minimizes lost time by returning to the steady state velocity once the hazard no longer exists.
   1.4. The system turns on when cruise control is enabled.
   1.5. The system contains a fail-safe mode, where hardware will adjust to maintain zero collisions in trade for increased lost time.
   1.6. The driver will be allowed to override the system at any time and regain full control over the vehicle.
   1.7. Pedestrian is moving at a right angle with respect to the vehicle.
   1.8. The driver will be notified when the vehicle begins to automatically decelerate upon detecting a potential collision or when the system fails.
   1.9. The system will always attempt to avoid collisions with pedestrians when decelerating the vehicle.

2. **Primary Requirements**

   2.1. **Software**
2.1.1. Continually monitor sensor detections from the Pedestrian Detection Sensor (PDS) which provides pedestrian recognition and tracking, pedestrian location relative to the vehicle, and pedestrian velocity to detect potential collisions between the vehicle and pedestrians.

2.1.1.1. Sends data collected by PDS (2.2 Hardware, 2.2.1) to controller every 100 m/s.

2.1.2. Controller receives data packets from the PDS, determining if there is a potential collision.

2.1.2.1. The controller can actuate the Brake-by-Wire (BBW) system to avoid a pedestrian collision.
2.1.2.2. When activating the BBW, the controller notifies the driver through speakers and a Heads-up Display.
2.1.2.3. Controller determines when there is no longer a potential collision, disabling BBW and returning the vehicle to a steady state velocity, reducing lost time.
2.1.2.4. The controller takes driver input when a potential collision is detected and the BBW system is actuated. If the controller detects the driver using a pedal or the brake lock, it will allow the driver to take control, disabling the BBW system.

2.2. Hardware

2.2.1. Pedestrian sensor is a stereo camera with the following detection properties.

2.2.1.1. Pedestrian location (x,y) relative to car with accuracy +/- .5 m.

2.2.1.2. Pedestrian velocity (speed and direction) Speed +/- .2 m/s. Direction +/- 5 degrees relative to vehicle path.

2.2.2. Brake-by-Wire (BBW) subsystem actuates vehicle brakes when directed by the controller.

2.2.2.1. Maximum of .7 g deceleration
2.2.2.2. Maximum 200 m/s response time to actuate after receiving a deceleration request.
2.2.3. There will be multiple driver indicators to communicate the state of the system.

2.2.3.1. Speakers in the vehicle will sound when a potential collision is detected and the BBW system is actuated. The speaker will sound until a potential collision is no longer detected.

2.2.3.2. A light indicator integrated with the vehicle’s Heads-up-Display (HUD) will light when a potential collision is detected and the BBW system is actuated. The light will remain until a potential collision is no longer detected.

2.3. Security

2.3.1. All components should contain a failsafe.
   2.3.1.1. When a component fails, the driver should be alerted to ensure driver safety and continue pedestrian detection.

2.3.2. The sensor, BBW, and controller will communicate through a hardwire connection, avoiding wireless communication.

2.3.3. Obfuscate the location of components to the driver, defending against modifications to the system.

4 Modeling Requirements

This section includes various models that describe the PCAS system. The models include a Use Case Diagram, Domain Model, Sequence Diagrams and State Diagrams. The components of each diagram are explained in detail.

4.1 Use Case Diagram

Figure 2 below shows the use case diagram for the PCAS. The stick figures represent the actors which consist of the Driver, Pedestrian, Pedestrian Sensor, Heads-up-Display, Speaker and Brake-by-Wire. The actors are placed outside of the system boundary, which is represented by the light blue box. Each actor has use cases that they interact with. The use cases are represented as blue ovals with lines connecting them to their actors. Dotted arrows are used to connect the use cases to each other. These are labeled as either includes or extends. An arrow with includes is
used to indicate that the use case being pointed at is included in the use case that is at the other end of the arrow. For example, Pedestrian Detected is included in Get Pedestrian Data. In order to get the location and speed of a Pedestrian, a Pedestrian must have been detected by the Pedestrian Sensor. An arrow with *extends* is used to indicate that the use case being pointed at is an extension of the use case at the other end of the arrow. For example, Avoiding a Collision is an extension of Decelerating. In the use case diagram, the PCAS forms the system boundary. Below the use case diagram are tables which provide a description of the use cases detailed in the figure above.

![Use Case Diagram](image-url)

Figure 2: Use Case Diagram
<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Turn on System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Driver</td>
</tr>
<tr>
<td>Description:</td>
<td>Once Cruise Control has been enabled, the PCAS turns on. While the system is on, the PCAS ensures there are zero collisions between the Vehicle and Pedestrians. The system requires no human intervention to achieve this.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes:</td>
<td>N/A</td>
</tr>
<tr>
<td>Extends:</td>
<td>N/A</td>
</tr>
<tr>
<td>Cross-refs:</td>
<td>Requirements 1.1, 1.2, 1.4</td>
</tr>
<tr>
<td>Use cases:</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Press Brake Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Driver</td>
</tr>
<tr>
<td>Description:</td>
<td>The Driver can press the Brake Lock of the Vehicle in order to override the PCAS. The Vehicle is no longer autonomous and is in manual mode with the Driver controlling the Vehicle.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes:</td>
<td>N/A</td>
</tr>
<tr>
<td>Extends:</td>
<td>Override System</td>
</tr>
<tr>
<td>Cross-refs:</td>
<td>Requirements 1.6, 2.1.2.4.</td>
</tr>
<tr>
<td>Use cases:</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Press Pedal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Driver</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Description:</td>
<td>The Driver can press the Pedal of the Vehicle in order to override the PCAS. The Vehicle is no longer autonomous and is in manual mode with the Driver controlling the Vehicle.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes:</td>
<td>N/A</td>
</tr>
<tr>
<td>Extends:</td>
<td>Override System</td>
</tr>
<tr>
<td>Cross-refs:</td>
<td>Requirements 1.6, 2.1.2.4</td>
</tr>
<tr>
<td>Use cases:</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Override System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Driver</td>
</tr>
<tr>
<td>Description:</td>
<td>While the PCAS is on, the Driver can override it by pressing the Brake Lock or the Pedal. While the PCAS is off, there is no monitoring and tracking of Pedestrians and it is completely up to the Driver to control the Vehicle to avoid colliding with a pedestrian.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes:</td>
<td>N/A</td>
</tr>
<tr>
<td>Extends:</td>
<td>N/A</td>
</tr>
<tr>
<td>Cross-refs:</td>
<td>Requirements 1.6</td>
</tr>
<tr>
<td>Use cases:</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>System Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Driver</td>
</tr>
<tr>
<td>Description:</td>
<td>When Cruise Control is not enabled, the PCAS will be off.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes:</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Use Case: Potential Collision Detected

**Actors:** Pedestrian

**Description:** Given the Pedestrian’s speed, direction and location along with the Vehicle’s speed, direction and location, the PCAS will determine whether or not there is a potential collision between the Pedestrian and Vehicle. The Pedestrian will always be moving at a right angle with respect to the Vehicle.

**Type:** Primary (essential)

**Includes:** Pedestrian Detected, Notify Driver, Decelerate

**Extends:** N/A

**Cross-ref:** Requirements 1.7, 2.1.1, 2.1.2, 2.2.1.1, 2.2.1.2

**Use cases:** Pedestrian Detected, Notify Driver, Decelerate

### Use Case: Pedestrian Detected

**Actors:** Pedestrian, Pedestrian Sensor

**Description:** The Pedestrian Sensor is continually tracking and monitoring any Pedestrian in its field of vision. If a Pedestrian exists in the field of vision, then it will be detected by the Pedestrian Sensor. This information is then sent to the PCAS to determine whether or not there is a potential collision between the Pedestrian and Vehicle given their locations, directions and speeds.

**Type:** Primary (essential)

**Includes:** N/A

**Extends:** N/A

**Cross-ref:** Requirements 2.1.1, 2.1.1.1

**Use cases:** Pedestrian Detected, Notify Driver, Decelerate
<table>
<thead>
<tr>
<th>Use cases</th>
<th>N/A</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Get Pedestrian Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Pedestrian Sensor</td>
</tr>
<tr>
<td>Description:</td>
<td>If the Pedestrian Sensor detects a Pedestrian, it then obtains the speed, x location, y location and direction of the Pedestrian. Once the Pedestrian Sensor has obtained this data, it sends the data to the PCAS. The PCAS then determines if there is a potential collision between a Vehicle and Pedestrian.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes:</td>
<td>Pedestrian Detected</td>
</tr>
<tr>
<td>Extends:</td>
<td>N/A</td>
</tr>
<tr>
<td>Cross-refs:</td>
<td>Requirements 2.1.2, 2.2.1.1, 2.2.1.2</td>
</tr>
<tr>
<td>Use cases:</td>
<td>Pedestrian Detected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Decelerate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Brake-by-Wire</td>
</tr>
<tr>
<td>Description:</td>
<td>When the PCAS determines that there is a potential collision between a Pedestrian and a Vehicle, a deceleration request is sent to the BBW system by the PCAS. The BBW then decelerates the Vehicle in order to avoid colliding with a pedestrian.</td>
</tr>
<tr>
<td>Type:</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes:</td>
<td>N/A</td>
</tr>
<tr>
<td>Extends:</td>
<td>Avoid Collision</td>
</tr>
<tr>
<td>Cross-refs:</td>
<td>Requirements 1.9, 2.1.2.1, 2.2.2, 2.2.2.1, 2.2.2.2, 2.3.2</td>
</tr>
<tr>
<td>Use cases:</td>
<td>Avoid Collision</td>
</tr>
<tr>
<td>Use Case</td>
<td>Avoid Collision</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Actors</td>
<td>Brake-by-Wire</td>
</tr>
<tr>
<td>Description</td>
<td>When the PCAS determines that there is a potential collision between a Pedestrian and a Vehicle, a deceleration request is sent to the BBW system by the PCAS. The BBW then decelerates the Vehicle in order to avoid colliding with a pedestrian. Once the potential collision has been mitigated, the BBW will stop braking and allow the Vehicle to resume its steady state speed.</td>
</tr>
<tr>
<td>Type</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes</td>
<td>N/A</td>
</tr>
<tr>
<td>Extends</td>
<td>N/A</td>
</tr>
<tr>
<td>Cross-refs</td>
<td>Requirements 1.1, 1.2, 1.3, 1.9, 2.1.2.1, 2.1.2.3, 2.2.2</td>
</tr>
<tr>
<td>Use cases</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Notify Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Heads-up-Display, Speaker</td>
</tr>
<tr>
<td>Description</td>
<td>When the PCAS determines that there is a potential collision between a Pedestrian and a Vehicle, a deceleration request is sent to the BBW system by the PCAS. The BBW then decelerates the Vehicle in order to avoid colliding with a pedestrian. Once the Vehicle begins decelerating, the Heads-up-Display and Speaker will visually and audibly notify the Driver of the deceleration. In addition, if the PCAS were to fail, the Heads-up-Display would notify the Driver of such a failure.</td>
</tr>
<tr>
<td>Type</td>
<td>Primary (essential)</td>
</tr>
<tr>
<td>Includes</td>
<td>N/A</td>
</tr>
<tr>
<td>Extends</td>
<td>N/A</td>
</tr>
<tr>
<td>Cross-refs</td>
<td>Requirements 1.5, 1.8, 2.1.2.2, 2.2.3, 2.2.3.1, 2.2.3.2, 2.3.1.1</td>
</tr>
<tr>
<td>Use cases</td>
<td>N/A</td>
</tr>
</tbody>
</table>
4.2 Domain Model

Figure 3 below shows a domain model for our system which depicts the key elements. We followed the UML class diagram notation so the boxes represent the key elements of our system. Inside of the box, the attributes are on top and the operations are beneath the attributes. For example, the PCAS has attributes such as systemOn and operations such as getSensorData(). The lines connected to the boxes represent relationships between the different elements in our system. One type of relationship depicted in the domain model is aggregation, where one element is a part of another element. This is represented by a line with a diamond. For example, the Wheel is part of the Vehicle. Another relationship is an association which is represented as a line between two boxes and a one to two-word description. For example, the PCAS communicates with the Brake-by-Wire. The last type of relationship depicted in the domain model is generalization. This is represented as a line with an arrow pointing to the base class and means the class at the other end of the arrow is an extension of the base class. For example, the Heads-up-Display is a DriverNotifier, just a special type of DriverNotifier. Each class in the domain model has a data dictionary listed below which details that class’ attributes, operations and relationships.
Figure 3: Domain Model for the PCAS
### Brake-by-Wire

**Description:** The Brake-by-Wire system receives deceleration requests from the PCAS. Once a deceleration request is received, the Brake-by-Wire system applies torque to the four Wheels of the Vehicle. Once the PCAS determines there is no potential collision, the Brake-by-Wire system stops applying torque and allows the Vehicle to return to steady-state velocity.

**Relationships**

<table>
<thead>
<tr>
<th>Associations:</th>
<th>PCAS, Wheel, Vehicle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregations:</td>
<td>Vehicle</td>
</tr>
<tr>
<td>Generalizations:</td>
<td>None</td>
</tr>
</tbody>
</table>

**Attributes:**

- torque: double - torque of the Brake-by-Wire system
- inFailSafeMode: bool - True if PCAS in fail-safe mode
- responseTime: int - time for responding to deceleration request

**Operations:**

- receivedDecelerationRequest(): bool - True if Brake-by-Wire system received a deceleration request from the PCAS
- overrideVehicleVelocity(): void - overrides the steady-state velocity of the Vehicle
- resumeVehicleVelocity(): void - stops overriding steady-state velocity of the Vehicle, Vehicle resumes steady-state speed

### Brake Lock

**Description:** A Vehicle has a Brake Lock. The Driver can hit the Brake Lock to override the PCAS.

**Relationships**

<table>
<thead>
<tr>
<th>Associations:</th>
<th>Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregations:</td>
<td>Vehicle</td>
</tr>
<tr>
<td>Generalizations:</td>
<td>Vehicle Component</td>
</tr>
</tbody>
</table>

**Attributes:** None

**Operations:** lock(): void - locks the Brake Lock
<table>
<thead>
<tr>
<th>Class</th>
<th>Description: The Driver is operating and controlling the Vehicle. The Driver is informed of potential collisions with Pedestrians. The Driver can hit either the Pedal or Brake Lock to override the PCAS.</th>
</tr>
</thead>
</table>
|               | **Relationships** | **Associations:** PCAS, Vehicle, Driver Notifier, Vehicle Component  
| Driver         | **Aggregations:** PCAS  
|                | **Generalizations:** None  
|                | **Attributes:**  
|                | - isManual: bool - True if Driver hits any input (either Pedal, Brake Lock) and overrides the PCAS  
|                | - cruiseActive: True when the Driver activates cruise control  
|                | **Operations:** None  
| Class          | Description: A Vehicle has a Driver Notifier which includes a Speaker and Heads-up-Display. The Driver Notifier lets the Driver know when the Vehicle is decelerating (due to a potential collision) or when the system is not working properly. |
| Driver Notifier | **Relationships** | **Associations:** PCAS, Driver  
|                | **Aggregations:** Vehicle  
|                | **Generalizations:** None  
|                | **Attributes:** None  
|                | **Operations:** notifyDriver(): notifies Driver of automatic deceleration and system failure  
| Class          | Description: A Heads-Up-Display is used to visually notify the Driver when the Vehicle begins to decelerate (due to a potential collision) or when the system is not working properly. |
| Class          | Description: A Heads-Up-Display is used to visually notify the Driver when the Vehicle begins to decelerate (due to a potential collision) or when the system is not working properly. |
# Heads-up-Display

<table>
<thead>
<tr>
<th>Relationships</th>
<th>Associations: PCAS, Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregations:</td>
<td>Vehicle</td>
</tr>
<tr>
<td>Generalizations:</td>
<td>Driver Notifier</td>
</tr>
</tbody>
</table>

**Attributes:**
- failureLight: bool - True if there is a system failure
- decelerationLight: bool - True if the Vehicle is decelerating due to a potential collision

**Operations:** None

---

## Class

### PCAS

**Description:** The main system that ensures there are zero collisions between a Pedestrian and Vehicle. The PCAS receives data from the Pedestrian Sensor and determines if there is a potential collision with a Pedestrian. If there is, the PCAS sends a deceleration request to the Brake-by-Wire and communicates with the DriverNotifer class in order to notify the Driver of the deceleration.

<table>
<thead>
<tr>
<th>Relationships</th>
<th>Associations: Driver, Pedestrian Sensor, Brake-by-Wire, Driver Notifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregations:</td>
<td>None</td>
</tr>
<tr>
<td>Generalizations:</td>
<td>None</td>
</tr>
</tbody>
</table>

**Attributes:**
- failSafeMode: bool - True if in fail-safe mode
- lostTime: double - lost time of the system
- systemOn: bool - True if system is on
- systemFailure: bool - True if there is a system failure
- systemOverridden: bool - True if Driver overrides system

**Operations:**
- enterFailSafeMode(): void - system enters fail-safe mode
- updateSystemStatus(): void - updates status of the system
- potentialCollision(): bool - True if there is a potential collision between a detected Pedestrian and the Vehicle
- sendDecelerationRequest(): message - sends a deceleration request to the Brake-by-Wire system
- getSensorData(): void - gets data from the Pedestrian Sensor
- endDecelerationRequest(): message - sends a message to the Brake-by-Wire system when the Vehicle no longer needs to decelerate

### Pedestrian

**Description:** Pedestrian is an external entity that could potentially collide with a Vehicle. Pedestrian’s speed and distance relative to the Vehicle are tracked and monitored by the Pedestrian Sensor and analyzed by the PCAS in order to detect potential Pedestrian-Vehicle collisions.

**Attributes:**
- initialX: int - initial X position of Pedestrian
- initialY: int - initial Y position of Pedestrian
- initialSpeed: int - initial speed of Pedestrian
- currentX: int - current X position of Pedestrian
- currentY: int - current Y position of Pedestrian
- currentSpeed: int - current speed of Pedestrian
- direction: int - direction of Pedestrian
- acceleration: int - acceleration of Pedestrian
- static: bool - True if Pedestrian is static

**Operations:**
- calculateXLocation(): int - updates X location of Pedestrian
- calculateYLocation(): int - updates Y location of Pedestrian

### Pedal

**Description:** A Vehicle has two Pedals. The Driver can hit either Pedal to override the PCAS.

**Relationships**
- **Associations:** Driver, Vehicle
- **Aggregations:** Vehicle
- **Generalizations:** Vehicle Component
### Class: Pedestrian Sensor

**Description:** The Pedestrian Sensor is a stereo camera that recognizes a Pedestrian and tracks their x location, y location, speed and direction relative to the Vehicle. The Pedestrian Sensor sends data to the PCAS so the PCAS can determine if there is a potential collision with a Pedestrian.

**Operations:**
- detectedPedestrian(): bool - True if the sensor detected a Pedestrian
- getPedestrianX(): int - gets X location of Pedestrian
- getPedestrianY(): int - gets Y location of Pedestrian
- getPedestrianSpeed(): int - gets speed of Pedestrian
- getPedestrianDirection(): int - gets direction of Pedestrian
- getVehicleX(): int - gets X location of Vehicle
- getVehicleY(): int - gets Y location of Vehicle
- getVehicleSpeed(): int - gets speed of Vehicle
- getVehicleDirection(): int - gets direction of Vehicle
- sendData(): data - sends Pedestrian and Vehicle location, speed and direction data to the PCAS
- sensorFailed(): void - Pedestrian Sensor lets the PCAS know that the Pedestrian Sensor failed

**Attributes:** None

**Relationships**
- **Associations:** Pedestrian, PCAS
- **Aggregations:** Vehicle
- **Generalizations:** None

---

### Class: Speaker

**Description:** The Speaker is responsible for notifying the Driver via a sound when the Vehicle begins to decelerate due to a potential collision detection.

**Relationships**
- **Associations:** PCAS, Driver
### Vehicle

**Description:** The Vehicle is an autonomous system that must never collide with a Pedestrian.

**Attributes:**
- STEADYSTATEVELOCITY: int - steady state velocity
- initialX: int - initial X position of Vehicle
- initialY: int - initial Y position of Vehicle
- currentX: int - current X position of Vehicle
- currentY: int - current Y position of Vehicle
- currentSpeed: int - current speed of Vehicle
- direction: int - direction of Vehicle

**Operations:**
- updateSpeed(): int - updates speed of the Vehicle

### Vehicle Component

**Description:** A Vehicle Component is a part of the Vehicle that the Driver can hit to override the PCAS. These include either Pedal and the Brake Lock.

**Relationships**

**Associations:** Driver

**Aggregations:** Vehicle

**Generalizations:** None
4.3 Sequence Diagrams

This section will discuss the sequence diagrams we used to illustrate how various components of our system will interact in different scenarios. Multiple scenarios are described in the following figures to capture the expected behavior of the system in both expected and unexpected situations. Sequence diagrams show each object as a box, its lifeline as a dotted line descending from that box, and messages between systems are displayed as arrows going from one object to the object it is communicating with. All these in conjunction give a representation of how the system will act in the given scenario.

4.3.1 Normal Driving, No Pedestrian Spotted

Figure 4 below shows the sequence diagram for the system's expected response when no pedestrians are detected by the sensor. Since no pedestrians are detected, the driver will proceed in a normal manner. As shown below, after the system activates, it will continuously scan for pedestrians, finding none. It will do this until the system is deactivated.
4.3.2 Pedestrian Detected, No Mitigation Needed

Figure 5 depicts the scenario in which a pedestrian is detected by the sensor, but is not a collision risk as the pedestrian is either too far away or moving away from the vehicle. As a result, the vehicle will proceed normally as mitigation is not necessary. As shown below, the system continuously scans for pedestrians. Once a pedestrian has been detected, the PCAS uses the data it receives from the sensors alongside the speed of the vehicle to determine if mitigation is necessary. In this example, it is not, so the vehicle proceeds as usual and goes back to tracking and monitoring pedestrians until the system is deactivated.
4.3.3 Pedestrian Detected, Mitigation Required

The scenario in Figure 6 occurs only when a pedestrian has been detected by the sensors, but this time the controller recognizes the pedestrian as a collision risk, and initiates the Brake-by-Wire to prevent a serious accident. Once the system is started, the sensors detect a pedestrian, the PCAS will use the data from the sensors and the speed of the vehicle to determine the pedestrian as a potential collision risk. At this time, the PCAS will communicate with the Heads-up-Display and Speaker to give audio and visual warnings to the driver, after which it will send information to the Brake-by-Wire to initiate braking in order to avoid a collision. Once the threat has passed, the vehicle resumes its steady state velocity.
4.3.4 Pedestrian Detected, Collision Mitigation Overridden

The scenario in Figure 7 is similar to the last, as once again a pedestrian has been detected and has been determined to be a collision risk. However, in this scenario, the driver overrides the PCAS by choosing to brake manually, at which point the Brake-by-Wire will stop its deceleration and give control to the driver.
4.3.5 Sensor Failure

In the scenario depicted in Figure 8, due to an issue with the sensor the system cannot safely operate as the data can’t correctly be given from the sensor to the controller. Upon recognizing this, the system will give a warning on the Heads-up-Display and deactivate until the system is repaired. At this point the driver is aware that the system is deactivated and must manually avoid collision threats until the system is once again operational.
4.4 State Diagrams

The following figures are a graphical representation of the objects in our system and how they will transition from one state to another, formally known as state diagrams. State diagrams build off of sequence diagrams to demonstrate a more low level idea about the objects in our system, specifically how these objects will behave under any given scenario. The notation is simple: each octangle represents a specific state an object in our class has the ability to be in. Arrows represent actions taken on those objects which then result in the state of the object changing. Additionally, each state diagram incorporates a circle with an arrow to indicate the starting state of each object.

4.4.1 Brake-by-Wire

Figure 9 represents the two states in which the Brake-by-Wire may transition between. The Brake-by-Wire will begin in the accepting state, where it will continually wait for deceleration requests. It will stay in this state until it receives a deceleration request. On this action, it will transition to the state where it adheres to this request and brakes the vehicle to avoid a collision. Once the collision is avoided, it will transition back to the accepting state, where it will then again wait for a deceleration request.
4.4.2 Cruise Control

Cruise Control enters the On state when the system is initialized. It will remain in the On state until the Driver manually turns Cruise Control off, in which case it will move to the Off state. When the Driver turns Cruise Control back on, it returns to its original On state.
4.4.3 Driver

The following figure depicts two of the states in which the Driver may encounter while interacting with the brake lock. Initially, the Driver is considered to be in an autonomous state: the boolean attribute isManual is set to false, indicating that the Pedestrian Collision Avoidance System (PCAS) is on. If the Driver pulls the brake lock, this action will transition the state of the Driver to a fully manual state, namely that the PCAS is shut down. Until the Driver releases the brake lock, it will remain in this state. Once the brake lock is released, the state transitions back to the original, in which the PCAS is on.
4.4.4 Driver

Similarly, the figure below depicts the two states the Driver will encounter when interacting with either pedal. Initially, the Driver is considered to be in an autonomous state; the boolean attribute isManual is set to false, indicating that the Pedestrian Collision Avoidance System (PCAS) is on. If the Driver presses either pedal, the state will transition to a fully manual state, turning the PCAS off. Once either pedal is released, the state will then transition back to the autonomous state, where the PCAS is on.
4.4.5 Driver

The state diagram in the following figure depicts another scenario for the Driver. In this case, the Driver starts in the steady state, where it is driving at a constant speed. If a pedestrian is picked up by the sensor, the Driver will notify the PCAS and ask if a collision is imminent. This action will transition to the PCAS, where if there is no imminent collision, it will merely transition back to the Driver's steady state. However, if a collision is imminent, it will notify the driver as well as send a declaration request to the Brake-by-Wire. The Driver Notifier will play an alert message while the system will cause the Driver to avoid a collision. Once this happens, the state of the Driver will be back in the steady state until another collision is imminent.
4.4.6 Heads-Up-Display

The following figure depicts the three states of the Heads-up-Display (HUD). The HUD begins in an idle state while the vehicle is in the Steady Speed state and no collisions or failures have occurred. When the PCAS detects a collision and begins deceleration, the deceleration light on the HUD is turned on and the HUD enters the Deceleration state. When deceleration is complete the light is turned off and the HUD reverts to Steady Speed. If the Pedestrian Sensor encounters a failure, the failure light is turned on and the HUD enters the Failure state. When the failure is cleared, the light is turned off and the HUD is once again in the Steady Speed state.
4.4.7 PCAS

The PCAS begins in the On state and will continue to retrieve data from the Pedestrian Sensor until either a Pedestrian is detected, there is a Pedestrian Sensor failure, or the system is turned off in which case it moves to the Off state. In the event of a Pedestrian Sensor failure, the PCAS moves to the Sensor Failure state and then turns on the failure light in the HUD before moving to the Off state until the system is turned back on. When a pedestrian is detected, the PCAS will calculate whether or not there will be a collision. If there is no collision it returns to the On state but if there is a potential collision it calculates the amount of braking needed in the Collision Detected state and sends a deceleration request to move into the Braking state. If there is any input from the Driver, it moves into the Override state until it is no longer receiving input. Otherwise, once the requested amount of deceleration has been met the PCAS returns to the On state and the cycle continues.
4.4.8 Pedestrian Sensor

The Pedestrian Sensor begins in the Getting Data state when the system is active. It will remain in the Getting Data state unless it encounters a failure, in which case it will enter the Failure state until the failure is cleared and the Pedestrian Sensor returns to its original state.

![State Diagram for the Pedestrian Sensor](image)

Figure 16: State Diagram for the Pedestrian Sensor

4.4.9 Vehicle

The Vehicle begins in the On state. When a call is made to getVehicleSpeed(), indicating that the PCAS has discovered a potential collision and action should be taken, the Vehicle moves to the Potential Collision state. If the Vehicle is able to brake in time and the collision is avoidable, the Vehicle moves back to On after its speed is adjusted with a call to updateSpeed(). Otherwise, in the case of an imminent collision, the Vehicle moves into the Imminent Collision state.

![State Diagram for the Vehicle](image)

Figure 17: State Diagram for the Vehicle
4.4.10 Vehicle Component

The vehicle component will begin in the Autonomous phase when the system is initiated. From there it will move to the Manual state if there is any input from the Driver. When Driver input has ceased, it will return to the Autonomous state until further Driver input is received.

![State Diagram for the Vehicle Component](Visual Paradigm Standard(Riley Thompson(Michigan State University)))

Figure 18: State Diagram for the Vehicle Component

5 Prototype

The intent behind this section is to introduce the prototype that we have built as a way to visualize and demonstrate the system requirements and behavior. The prototype was developed with HTML, CSS, and Javascript.

5.1 How to Run Prototype

The PCAS prototype is accessible via our team website [2] or the following link: https://distracted-benz-5b1790.netlify.app/. To begin, simply select a scenario and click Start Simulation. As the simulation progresses, any alerts will appear in the Alert Log just below the simulation window. You can also monitor the state of select subsystems under the Subsystems section toward the bottom of the screen.
5.2 Sample Scenarios

This section outlines a scenario the prototype may encounter that illustrates how the PCAS should behave under certain conditions. A description of the scenario along with an image depicting the corresponding system behavior is provided.

All scenarios begin with the vehicle traveling at a steady speed and a pedestrian at the side of the road.

In Figure 19, notice the empty Alert Log as the vehicle has yet to detect the pedestrian. Additionally, notice the Subsystems where the system is active and the Brake-by-Wire and Driver HUD is inactive.

5.2.1 Sample Scenario 1

Scenario 1 - The vehicle is traveling at a steady speed and a pedestrian starts to move into the vehicle path. The sensor detects the pedestrian moving in the vehicle path and determines a potential collision. When a potential collision is detected, the system activates the Brake-by-Wire.
In Figure 20, notice that the vehicle is stopped with ample room for the pedestrian. Notice under the Alert Log that a pedestrian was detected and the brakes were applied. Also notice under Subsystems the Brake-by-Wire subsystem is active and the Driver HUD is active.

6 References


7 Point of Contact

For further information regarding this document and project, please contact Prof. Betty H.C. Cheng at Michigan State University (chengb@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.