Software Requirements Specification (SRS)

Automated Pedestrian Collision Avoidance System (APCA)

Authors: Team GReEN; Garret Smith, Rebecca Collins, Eric Austin, Nikhil Andrews

Customer: Mr. David Agnew, Continental Automotive Systems

Instructor: Dr. Betty Cheng, Michigan State University

1 Introduction

This software requirements specification (SRS) document provides an overview of the document's purpose, scope, definitions, acronyms, abbreviations, and the overall organization of the product.
Section 2 gives the overall description of the Automated Pedestrian Collision Avoidance (APCA) system. This section includes the product perspective, product function, user characteristics, constraints, assumptions and dependencies, and apportioning of requirements.
Section 3 provides the requirements and Section 4 provides the models of our system. Section 5 contains a demonstration of the prototype, details on how to run the prototype, and sample scenarios. Section 6 identifies references for this document. Finally, Section 7 explains the point of contact for further information.

1.1 Purpose

The purpose of the SRS is to describe the specifics of the APCA System; how the system behaves, system goals, specifications and constraints under which the system operates. This document is mainly intended for those developers working on such systems and customers of the APCA system, but in our case is more specifically intended for Mr. David Agnew of Continental Automotive Systems.

1.2 Scope

The Automated Pedestrian Collision Avoidance System (APCA) is a fully autonomous system that is designed to avoid collisions with pedestrians. This system provides an automated control over the vehicle's braking system (BBW or Brake-By-Wire) that responds to potential collisions while minimizing loss of efficiency (time). The system also has control over the vehicle's acceleration in order to return to original speed after avoiding collision. Overall, the APCA system is designed to provide additional safety to the driver and pedestrians.

1.3 Definitions, acronyms, and abbreviations

This section of the SRS contains definitions, acronyms, and abbreviations for the terminology used to describe our system throughout this document.

- **APCA** – Automated Pedestrian Collision Avoidance
- **State** – The condition the system is in.
- **Pedestrian Sensor** – A stereo camera that provides pedestrian recognition, relative location, speed, and relative direction with respect to the vehicle.
- **Packet** – A piece of data generated by the pedestrian sensor containing the relative locations, speeds, and relative directions of pedestrians in and near the path of the vehicle.
- **Cycle Time** – the time it takes for the pedestrian sensor to send the packet of signals. In the project overview, it is given as 100 ms.
- **Brake-by-Wire (BBW)** – a sub-system that responds to deceleration requests by interrupting the steady speed (set by cruise control) and then applying brakes.
- **Vehicle** – the autonomous automobile for this application
- **“Lost time”** – time difference (in seconds) between system on (vehicle starts to avoid collision) and system off (vehicle avoids and returns back to steady state speed).
- **Fail Safe** – or fail operational mode, increases the response time to apply brakes in order to reach the requested deceleration. The response time to reach the requested deceleration increases from 200 ms to 900 ms. This means that more time is allowed for braking to ensure collisions are avoided. By beginning to brake sooner, the system is compensating for hardware that may not be functioning properly.

1.4 Organization

The remainder of this document is organized as follows.
Section 2 of this document, Overall Description, gives a general idea of how the product functions. Some of the key concepts described are: interface constraints, the intended users, and assumptions.
Section 3 of this document, Specific Requirements, gives an enumerated list of requirements that the system fulfills.

Section 4 of this document, Modeling Requirements, explains how the software is designed to meet the requirements. In this section, UML diagrams used to specify how the software functions.
Section 5, Prototype, provides a demonstration on how to use and run the prototype and includes sample scenarios of the system.
Section 6, References, gives a list of all documents referenced.
Section 7 of this document, point of contact, gives information on how to obtain more information regarding this document and project.

2 Overall Description

Autonomous driving is an area of great interest to the auto industry. An essential task of an automated car is to avoid collisions. This section of the document will give an overview of the APCA system. Section 2.1 will provide a product perspective for the reader, providing a high level description of the product. The next section, Section 2.2, will describe the products functions in more detail. Section 2.3 gives characteristics of the products users. Section 2.4 details the constraints for the project, and Section 2.5 addresses the assumptions that were made and the projects dependencies. Finally, Section 2.6 discusses additional requirements that were beyond the scope of the current project.

2.1 Product Perspective

The APCA system is an autonomous safety system used for avoiding collisions with pedestrians. The system detects a pedestrian with a stereo camera and tracks their location and path. If a collision is imminent, a signal is sent to the BBW actuator to begin braking. As shown in the data flow diagram in Figure 1, the system interfaces with the BBW actuator and with the stereo camera used for pedestrian detection. Since the vehicle is autonomous, no user interface exists for this system. The system is active whenever the vehicle is on, and receives a signal from the stereo camera every 100ms.

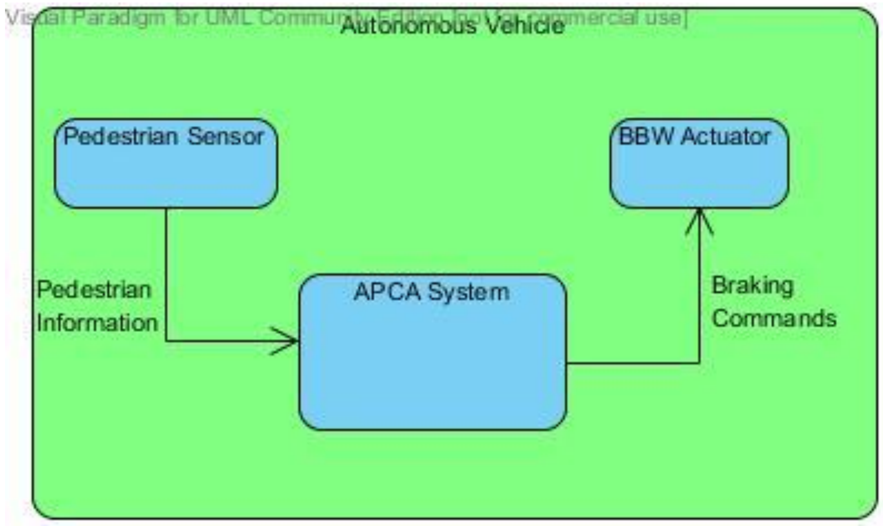


Figure 1

2.2 Product Functions

The following are functions of our product as specified by the customer.

- Detection: The use of the pedestrian stereo camera to locate pedestrians in front of the vehicle.
- Tracking: Using the camera and algorithms, track the position of the pedestrian as it moves within the sensors scanning range.
- Apply Braking: When a pedestrian steps in front of the vehicle and a collision is possible, a signal is sent to the BBW actuator to begin deceleration.
- Restore Velocity: When either the pedestrian threat is gone or braking has been applied, signal the vehicle to restore its previous velocity.
- Fail Safe: If the vehicle detects a problem in the braking hardware, the fail safe mode is enacted. The time to reach the requested deceleration increases from 200 ms to 900 ms, and more time is allowed for braking to ensure collisions are avoided.

The project description also described 2 high level goals for the system.

- Safety Effectiveness: There shall be zero vehicle/pedestrian collisions for each scenario
- Efficiency: Minimize "Lost Time" due to safety maneuvers. This must not interfere with safety effectiveness.

2.3 User Characteristics

The user in the context of this product would be considered the driver of the vehicle. However, the vehicles that will be equipped with the APCA system will be fully autonomous. Therefore, the driver is assumed to have little or no knowledge of the system. The driver is also expected to have no interaction with the system.

2.4 Constraints

The APCA system must have zero collisions for each scenario presented as stated in the Project Description. The system must also attempt to avoid “lost time” due to pedestrian avoidance. This must not interfere with safety constraints. If a braking problem is present, the APCA system must enable the fail safe system and increase time allowed for braking. Additionally, the BBW actuator is configured to respond to commands as quickly as a human. The response time to reach the requested deceleration is 200ms and the brake release time is 100ms. Also, with the given scenarios, the vehicle will always be travelling at a steady velocity of 13.9 m/s and have an acceleration of 2.4525 m/s^2 after braking is applied.

2.5 Assumptions and Dependencies

Some assumptions were clarified by the customer about the system and elements that interact with it. It is assumed that the APCA system is always on and scanning if the vehicle is running. The pedestrian is assumed to accelerate instantly and move at either 0 or 6 kilometers per hour. Also, the pedestrian will always be moving at a 90 degree angle to the vehicles path. It is also assumed that the elements of the system are all functioning properly and no abnormal conditions are present.

2.6 Approportioning of Requirements

Based on customer negotiations, some features are beyond the scope of our project and may be addressed in the future. One such feature is tracking multiple pedestrians. The current product need only worry about tracking one pedestrian at a time. Also, erratic pedestrian movement will not be a factor in our product. The pedestrian will always move at a steady speed. Additionally, since the vehicle is expected to be fully automated, the system has no need to interact with the driver.

3 Specific Requirements

1. **The system consists of a pedestrian sensor and a BBW actuator.**
   a. The pedestrian sensor will be a stereo camera and will send a packet every 100 ms. The data in the packet contains the pedestrian location (+/- 0.5 m) relative to the car, the pedestrian speed (+/- 0.2 m/s), and direction (+/- 5 deg).
   b. The Brake-by-Wire (BBW), when activated, interrupts the steady state velocity control (cruise control) and applies braking torque at all four wheels of the vehicle. It has a deceleration accuracy of +/- 2%, a response time of 200ms to reach requested deceleration, and a response time of 100 ms to release. A maximum deceleration of 6.867 m/s^2 is possible.
2. The vehicle and the pedestrian for testing the system have the following properties:
   a. The vehicle is an autonomous vehicle that has a normal steady state speed of 13.9 m/s and an acceleration back to steady state speed (after auto brake apply) of 2.4525 m/s^2. The collision zone is based on the vehicle's width of 2 m.
   b. The pedestrian for this application can be in static or in motion and have speeds of 0 m/s OR 1.67 m/s and it can be assumed that the pedestrian can accelerate instantly. The size of the pedestrian can be considered a circle with 0.5 m diameter.
3. The system sensor should be able to calculate the distance required to brake between the vehicle and pedestrian. The system should then adjust deceleration in order to avoid an accident with the pedestrian.
4. The system must return to steady-state velocity after the auto braking maneuver with an acceleration of 2.4525 m/s^2.
5. When the vehicle is in a potential collision zone, the system should take action immediately to brake and avoid collisions.
6. Based on the scenarios described in the project overview, the system should react appropriately no matter which scenario is being simulated.
7. The system should be effective (zero collisions allowed) and efficient (minimize lost time).

4 Modeling Requirements

Figure 2 shows the different interactions available to users of the APCA system. The stick figure of a person symbolizes a user or other external actor, and the arrow that protrudes from them reach ovals that represent high-level actions that a user can take to affect the system's behavior. The <<Include>> label symbolizes that the use case being pointed to is dependent on the other use case.

Use Case Diagram

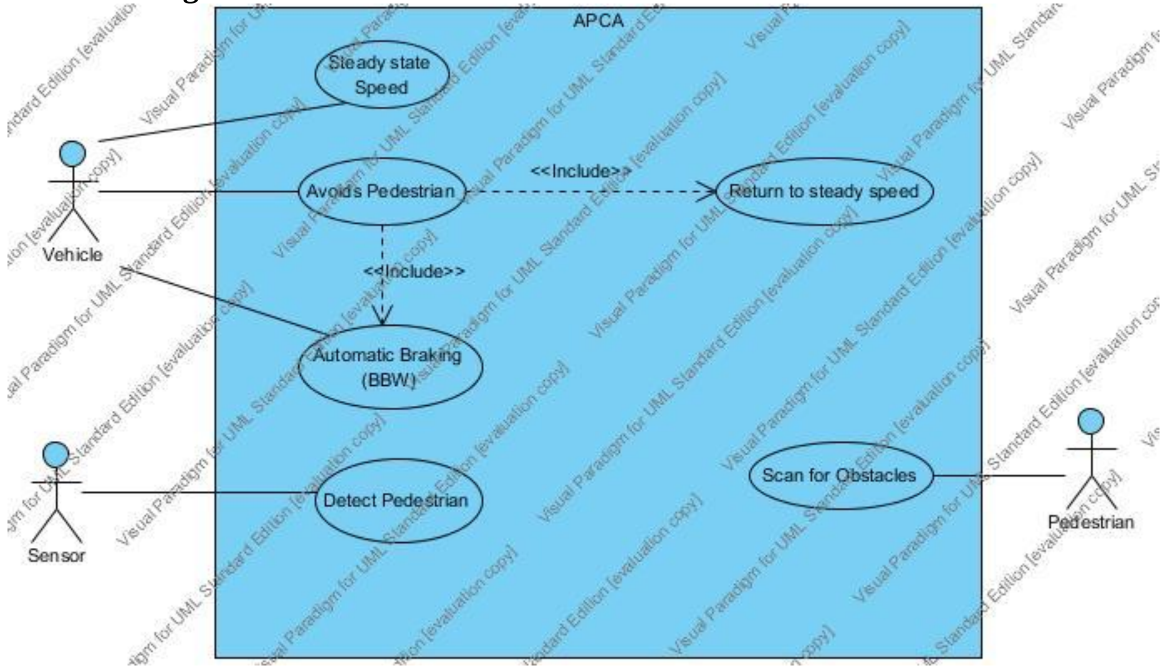


Figure 2

Use Case:	Detect Pedestrian
Actors:	Sensor
Type:	Primary and Essential
Description:	Sensor scans, recognizes and tracks pedestrians in front of it.
Cross-references:	1a, 3, 6, 7

Use Case:	Steady State Speed
Actors:	Vehicle
Type:	Primary and Essential
Description:	The Autonomous Vehicle is set to a steady velocity.
Cross-references:	2a, 6, 7

Use Case:	Avoids Pedestrian
Actors:	Vehicle
Type:	Primary and Essential
Description:	The Autonomous Vehicle moves in forward direction, the sensor detects the pedestrian and avoids collision.
Includes:	Automatic Braking (BBW), Return to Steady Speed
Cross-references:	2a, 4, 5, 6, 7
Related Use-Cases:	Activates Automatic Braking (BBW) and Return to Steady State (if necessary) use cases

Use Case:	Return to Steady Speed
Actors:	Sensor
Type:	Primary and Essential
Description:	The Sensor detects a pedestrian, the vehicle brakes and avoids collision. Then, the vehicle resumes back to the original set speed.
Cross-references:	2a, 4, 6, 7

Use Case:	Automatic Braking (BBW)
Actors:	Vehicle
Type:	Primary and Essential
Description:	Once the pedestrian has been detected, the BBW sub-system responds to deceleration requests and automatically starts braking accordingly until collision has been avoided.
Cross-references:	1b, 3, 5, 6, 7

Use Case:	Scan for Obstacles
Actors:	Pedestrian
Type:	Primary and Essential
Description:	The pedestrian can be in static or in motion, can change speed with infinite acceleration. When moving, the pedestrian only moves in right angle to the vehicles path.
Cross-references:	2b, 6, 7

Class Diagram and Data Dictionary

The class diagram in Figure 3 shows the structure of the APCA system by displaying the system's classes, their attributes, operations, and the relationships among objects. Associations between these classes are shown by solid lines. Each association has a text description which shows how the classes relate. They also have multiplicity which describe how many objects of each class are involved in the relation ("*" means "any"). Aggregation is shown like a regular association, but with a diamond on the line. In a relationship, the class that has the diamond touching it contains the other class. The Pedestrian Sensor, which is a subsystem of the APCA System, will constantly scan for pedestrians. The APCA System and the BBW Actuator are both subsystems of the Vehicle, and will work together to prevent the Vehicle from colliding with a Pedestrian. The fail safe mode will be activated if a hardware failure occurs. To achieve this, the Vehicle will use the AlertOfFailSafe() function to communicate the problem with the APCA System. If the APCA System is in fail safe mode, the vehicle will take longer to decelerate.

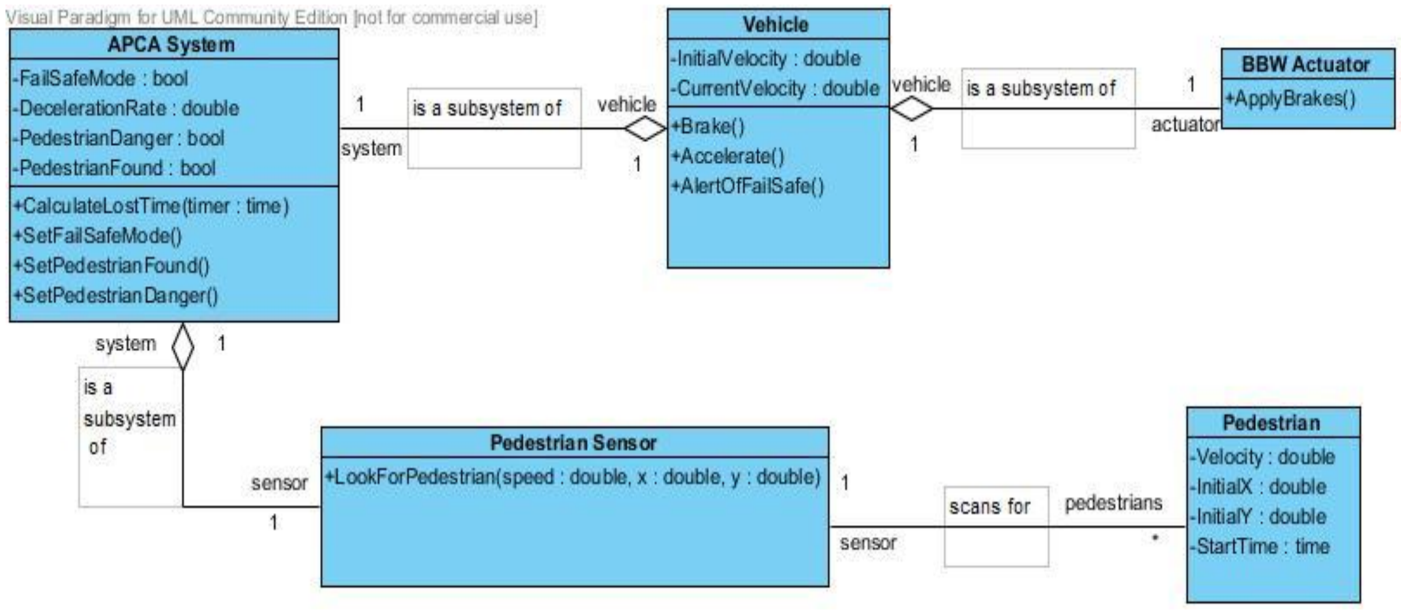


Figure 3

Below are several tables that make up the data dictionary. Theses tables show detail about the attributes, operations, and relationships of the classes within our project.

Element Name		**Description**
APCA System		Brief description (e.g., purpose and scope).
Attributes		
	DecelerationRate: double	Shows how fast the vehicle can decelerate.
	FailSafeMode: boolean	Shows if the system is in fail safe mode or not.
	PedestrianDanger: boolean	Shows if the pedestrian is in danger of collision.
	PedestrianFound: boolean	Shows if a pedestrian has been found by the system.
Operations		
	CalculateLostTime (Timer : time): void	Calculates how much time is lost if and when the vehicle decelerates to avoid a pedestrian.
Relationships	The APCA System is a subsystem of the Vehicle.	
UML Extensions		

Element Name		**Description**
BBW Actuator		Brief description (e.g., purpose and scope).
Attributes		
Operations		
	ApplyBrakes (): void	If the vehicle decides to brake, the BBW Actuator will use this operation to slow the vehicle.
Relationships	The BBW Actuator is a subsystem of the Vehicle.	
UML Extensions		

Element Name		**Description**
Pedestrian		Brief description (e.g., purpose and scope).
Attributes		
	InitialX: double	The starting point of the pedestrian in regards to the X axis.
	InitialY: double	The starting point of the pedestrian in regards to the Y axis.
	StartTime: time	The time at which the pedestrian will start walking across the street.
	Velocity: double	The velocity at which the pedestrian is walking.
Operations		
Relationships	The Pedestrian triggers the Pedestrian Sensor.	
UML Extensions		

Element Name		**Description**
Pedestrian Sensor		The Pedestrian Sensor scans for pedestrians that the
Attributes		
Operations		
	LookForPedestrian (Speed : double, X : double, Y : double): void	The Sensor will constantly scan for pedestrians that the vehicle may collide with.
Relationships	The Pedestrian Sensor is a subsystem of the APCA System. It scans for Pedestrians that the Vehicle may potentially collide with.	
UML Extensions		

Element Name		**Description**
Vehicle		Brief description (e.g., purpose and scope).
Attributes		
	CurrentVelocity: double	The current velocity the vehicle is at.
	Velocity: double	The initial velocity of the vehicle.
Operations		
	Brake (): void	Sends a signal to the BBW actuator to begin applying brakes.
	Accelerate (): void	The vehicle accelerates back to its initial velocity.
	AlertOfFailSafe (): void	Alerts the system of the failsafe state.
Relationships	The Vehicle contains both the APCA System and the BBW Actuator	
UML Extensions		

State Diagram

Below in Figure 4 is a state diagram for the APCA system. The big bullet points in the diagrams show the system's beginning state when the scenarios begin. The blue boxes within each classes diagram are states the system can be at a given time. The arrows between the boxes display how transitions can be made from one state to another. Additionally, the text on the arrows shows operations that are performed during the transition, and the text in brackets shows the conditions for this state transition to occur. The system constantly scans for pedestrians as soon as the vehicle is turned on. For each new pedestrian found, it begins tracking. When a pedestrian steps in front of the car, it is seen as being in danger of collision. A signal is then sent to the BBW actuator and braking is applied. As soon as the pedestrian is out of danger, acceleration begins to bring the vehicle up to speed. During acceleration, the tracking module makes sure no pedestrians enter a collision course. After initial velocity is achieved again, the system calculates time lost due to evasive maneuvers.

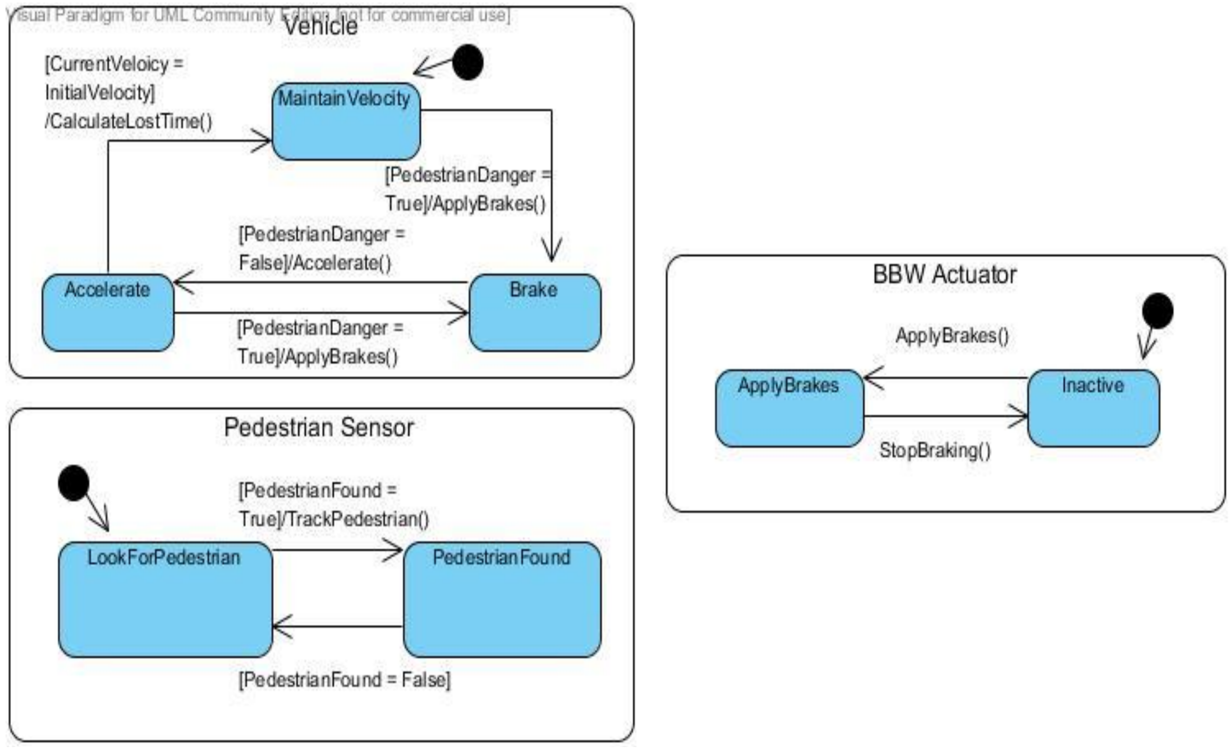

Visual Paradigm for UML Community Edition [not for commercial use]
Vehicle
[CurrentVeloicy = InitialVelocity] /CalculateLostTime()
MaintainVelocity
[PedestrianDanger = True]/ApplyBrakes()
[PedestrianDanger = False]/Accelerate()
Accelerate
Brake
[PedestrianDanger = True]/ApplyBrakes()
BBW Actuator
ApplyBrakes()
ApplyBrakes
Inactive
StopBraking()
Pedestrian Sensor
[PedestrianFound = True]/TrackPedestrian()
LookForPedestrian
PedestrianFound
[PedestrianFound = False]

Figure 4

Sequence Diagrams

Below are sequence diagrams for different scenarios the system may encounter. The blue boxes at the top of the diagrams are the class objects being used in a scenario. The dotted line under them represents the timeline for that object. The Actor is an external object that is interacting with the objects in the scenario. Arrows travel from the Actor to the objects to symbolize the initiation of function calls within the objects. Arrows pointing back towards the function caller represent the return values. This allows the viewer to see how different data returns can result in different program flows. Figure 5 shows a scenario in which no pedestrian present. The vehicle maintains its current velocity and continues as normal. The second scenario shown in Figure 6 is with a pedestrian present. When the pedestrian is in danger of being in a collision, the vehicle applies braking until the danger has passed. Then the vehicle accelerates back up to its initial velocity and continues.

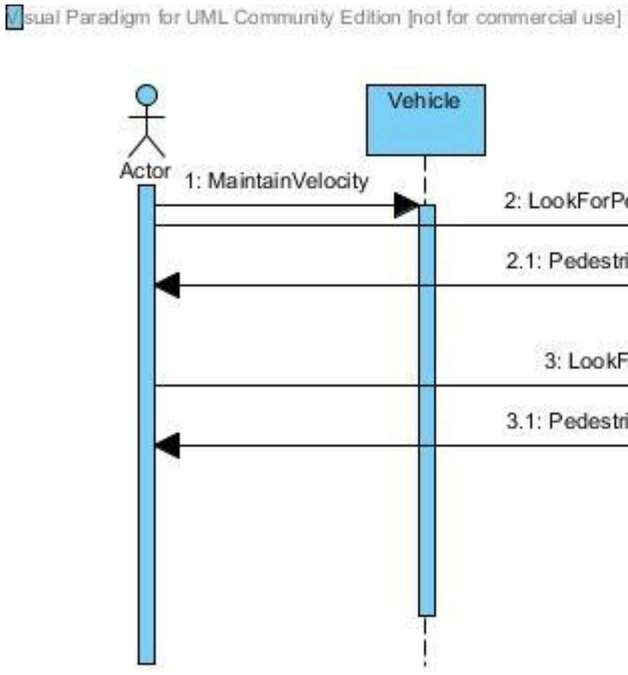

Visual Paradigm for UML Community Edition [not for commercial use]
Actor
Vehicle
Pedestrian Sensor
1: MaintainVelocity
2: LookForPedestrian()
2.1: PedestrianFound = False
3: LookForPedestrian()
3.1: PedestrianFound = False

Figure 5

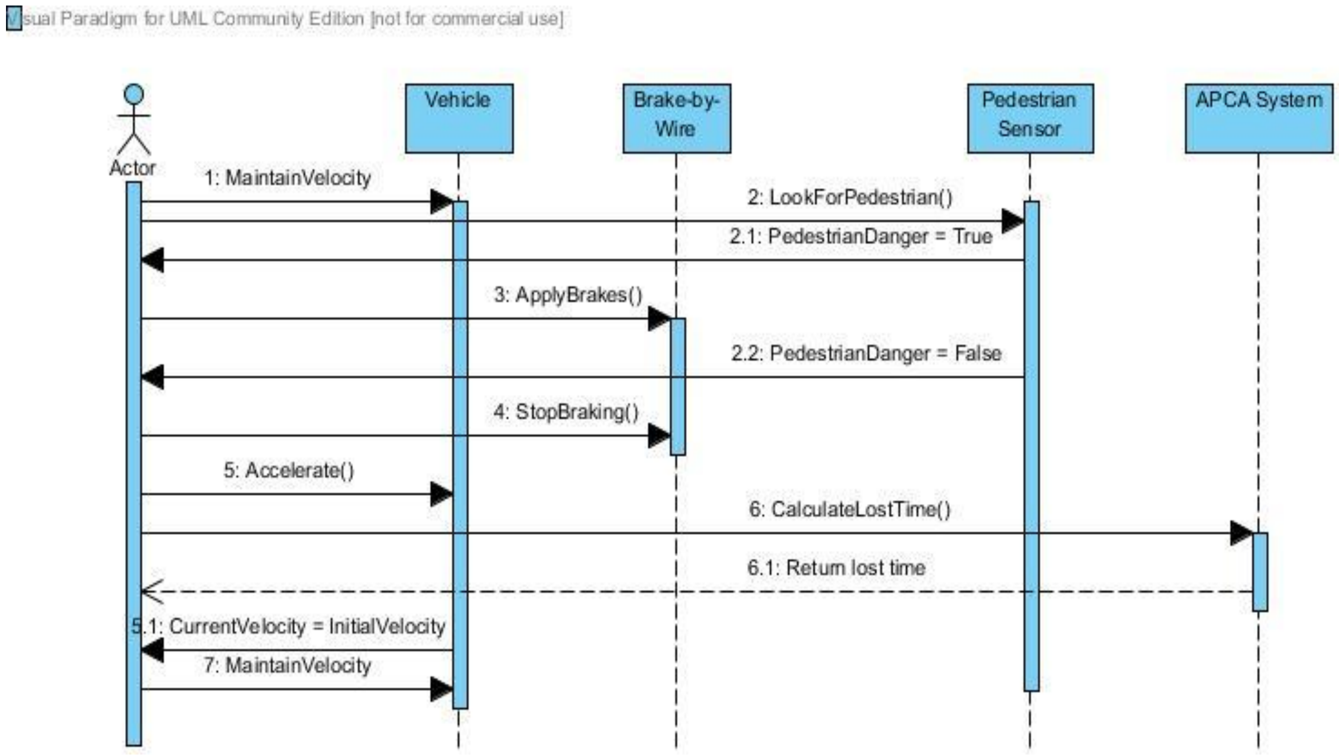


Figure 6

5 Prototype

The APCA System itself does not contain any sort of user interface. Since the system is run by a fully autonomous vehicle, the prototype is a scenario simulator. The scenarios in the project specifications gave details on how the system should respond to them. Our prototype will allow users to simulate one of the predefined scenarios to run, or supply their own scenario and run it. The results will be displayed in a table and the amount of time lost will be displayed.

5.1 How to Run Prototype

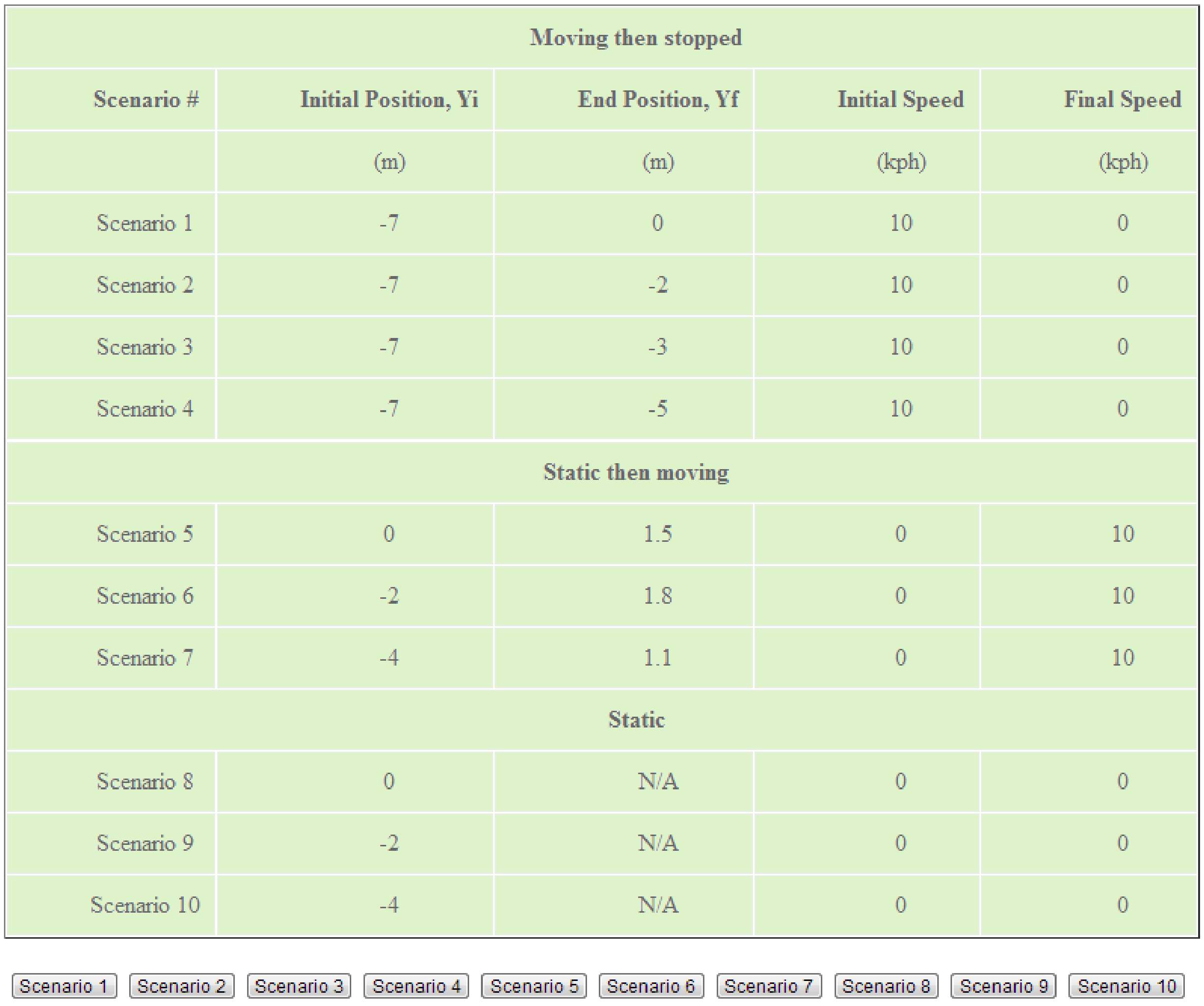
To run our prototype, all you need is a web browser (Google Chrome, Mozilla Firefox, Safari, Opera, Internet Explorer, etc.) that supports JavaScript.

To access the prototype follow this URL link:
http://www.cse.msu.edu/~cse435/Projects/F2013/Groups/APCA2/web/prototype.html

The prototype currently does not execute, but following is an example of how it will work. The customer arrives to our webpage and is presented with a table containing all the information of the predefined scenarios we will test for. Beneath the table are 10 buttons, one for each predefined scenario. Upon clicking the button, we will run our algorithm on the variables to determine if the pedestrian is hit or not, and what the loss of time for each scenario will be. This information will be displayed in the "Results" box at the bottom of the page. There is also a form that allows for custom scenarios to be run. The customer will be able to plug in numbers to the text boxes and click the "Compute Scenario" button to test the system under any conditions.

5.2 Sample Scenarios

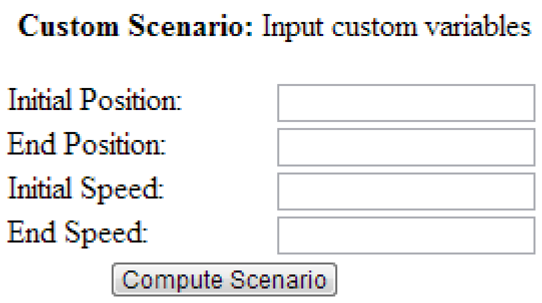
The table below, Figure 7, displays the details of the predefined scenarios is available for users to view. They can then click one of the 10 buttons to compute one of those scenarios.



Moving then stopped				
Scenario #	Initial Position, Yi	End Position, Yf	Initial Speed	Final Speed
	(m)	(m)	(kph)	(kph)
Scenario 1	-7	0	10	0
Scenario 2	-7	-2	10	0
Scenario 3	-7	-3	10	0
Scenario 4	-7	-5	10	0
Static then moving				
Scenario 5	0	1.5	0	10
Scenario 6	-2	1.8	0	10
Scenario 7	-4	1.1	0	10
Static				
Scenario 8	0	N/A	0	0
Scenario 9	-2	N/A	0	0
Scenario 10	-4	N/A	0	0

Figure 7

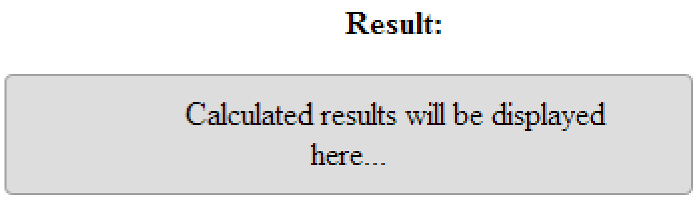
If the user wants to test a custom scenario, one that has details different from the predefined scenarios, they can use the form to compute a user defined scenario.



Custom Scenario: Input custom variables

Initial Position:	
End Position:	
Initial Speed:	
End Speed:	

After a scenario has been computed, the results of the scenario and the amount of time lost will be calculated and displayed in the Results Box.



Calculated results will be displayed here...

6 References

[1] CSE435-Team GReEN, Project Website,
http://www.cse.msu.edu/~cse435/Projects/F2013/Groups/APCA2/web/

[2] Continental Automotive, Functional Algorithm for Automated Pedestrian Collision Avoidance System, Online, Available:
http://www.cse.msu.edu/~cse435/Projects/F2013/ProjectDescriptions/APCA-2013.pdf

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.