<Front Page>

**Software Architecture Document**

iDrive

Delta Dynamos

Sassan Estrada

Samuel Flinkfelt

Efrain Bonilla Caudillo

Luc Dang

Anthony Goosen

Gunjan Gupta

Blachly Hunter

Presentation and Due Date: Presentation – Part II Dec 8

**SAD Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 9/22 | 1 | Introduction | Sam |
| 9/25 | 2 | Logical View | Anthony / Sam |
| 10/1 | 3 | Architecturally Significant  Design Packages | Sassan/Efrain |
| 10/03 | 4 | Implementation/ Development View | Anthony / Sam |
| 10/06 | 5 | Architecturally Significant Design Classes | Luc / Hunter |
| 10/12 | 6 | Deployment/Physical View | Sassan/Efrain |
| 10/13 | 7 | Process View | Luc / Hunter |
| 10/14 | 8 | Architectural Representation | Luc |
| 10/17 | 9 | Quality | Sassan |
| 10/17 | 10 | Architectural Goals and Constraints | Efrain |
| 12/2 | 11 | Edits from feedback | Efrain |
| 12/4 | 12 | Edits from feedback | Hunter |

Table of Contents

[1. **INTRODUCTION 4**](#kix.k15boqcf7gcn)

[1.1 PURPOSE 4](#kix.d478ofns8uw6)

[1.2 SCOPE 5](#kix.2mb36n2k0g9v)

[1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS 5](#kix.tcz2sr2x2nmm)

[1.4 REFERENCES 5](#kix.c6x3ft74ei7t)

[1.5 OVERVIEW 6](#kix.hs7bh8huin7f)

[**2.** **ARCHITECTURAL REPRESENTATION 7**](#kix.2pkh0ri8gt02)

[**3. ARCHITECTURAL GOALS AND CONSTRAINTS 7**](#kix.3so1u79emccz)

[**4. USE-CASE VIEW 8**](#kix.meugg13lr50z)

[**5. LOGICAL VIEW 16**](#kix.y13wxhw2y85f)

[5.1 OVERVIEW 18](#kix.nyetfqvq1z6v)

[5.2 ARCHITECTURALLY SIGNIFICANT DESIGN PACKAGES 18](#kix.6m876nprb8ak)

[5.3 ARCHITECTURALLY SIGNIFICANT DESIGN CLASSES 23](#kix.q54avmfswc72)

[**6. IMPLEMENTATION/DEVELOPMENT VIEW 27**](#kix.qqt16da2lp9t)

[**7. PROCESS VIEW 28**](#kix.jgncqk89b9kn)

[**8. DEPLOYMENT/PHYSICAL VIEW 32**](#kix.b87s2nf5rykw)

[**9. Size and Performance 32**](#kix.5bbg2r2f8bkk)

[**10. Quality 33**](#kix.ndtktj8vqss1)

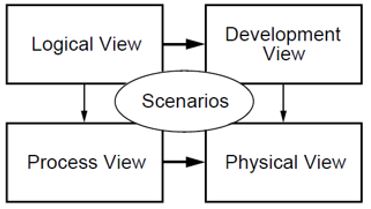
**1. Introduction**

iDrive is a simple driver assistance system which performs Speed Control System (SCS), Steering automation with Lane detection, Obstacle avoidance detection along with GPS capabilities to show exactly where and how fast the car is going. This system is growing to support a wide range of car makes, models and years and is easy to set up on vehicles which includes a fully functioning self driving assistance system. It uses the onboard OBD II system integrated with sensors and self driver monitoring camera system to control the aspects of driving to make the most informed decisions. This gives full control of the vehicle along with all the sensors giving it the utmost responsive behavior to avoid accidents and detect normal driving conditions for driving the vehicle. This system architecture is abstracted into many views and breaks down the components that are explained in detail. This document follows the 4+1 model as the reference model for this document.

**1.1.** **Purpose**

This document contains the descriptions of the system, in terms of a 4+1 view model which displays various architectural views. This serves as a communication medium between the software architect and other project team members, regarding architecturally significant decisions that will be made on the project.

4+1 View Model

**

**1.2. Scope**

The scope of this project will be within the scope of creating a self driving system, with all it’s components that can work integrated with all it’s hardware and make decisive decisions with the safety of the driver as it’s top priority. These will include sensors that will be integrated with the system, such as a camera detection and monitoring system, speed sensors, gps, and speed sensing devices, gyroscope and computer system.

**1.3**  **Definitions, Acronyms, and Abbreviations -**

SRS - System Requirements Specification

OBDII - On-Board Diagnostic system

GPS - Global Positioning System

UML - Unified Modeling Language

SAD - Software Architecture Document

**1.4. References -**

**Use Case**

Affairs, A. (2013, October 09). Use Cases. Retrieved October 19, 2020, from https://www.usability.gov/how-to-and-tools/methods/use-cases.html

Use Case Diagram Guidelines for Better Use Cases. (2018, September 26). Retrieved September 29, 2020, from https://creately.com/blog/diagrams/use-case-diagram-guidelines/

Mahachi, T. C. (2018, July 10). 5 Rules for Writing Effective Use Cases :: Systems Flow, Inc. Retrieved October 10, 2020, from http://www.sysflow.com/blog/5-rules-for-writing-effective-use-cases/

**Architecture**

(SRS) System Requirement Specification Document

Software Architecture in Practice. (n.d.). Retrieved October 13, 2020, from https://sites.google.com/site/softwarearchitectureinpractice/9-documenting-software-architecture/d-allocation-views/a-deployment-view

**UML Diagram**

(n.d.). Retrieved from http://draw.io/

(n.d.). Retrieved from https://www.lucidchart.com/pages/

**1.5. Overview -**

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the representation of the 4+1 model the system

Section 3: describes the Goals and Constraints of the project

Section 4: describes the functional requirements with a significant impact on the architecture

Section 5: describes the Logical overview of the system architecture

Section 6: describes the Implementation and Development overview of the system and its components

Section 7: describes how the systems sequence certain functions which includes process views

Section 8: describes the physical layers and subsystems of the application

Section 9: describes the performance issues and size of the system

Section 10: describes the scalability, reliability and availability, security. This incorporates a broad overview of the quality of the system

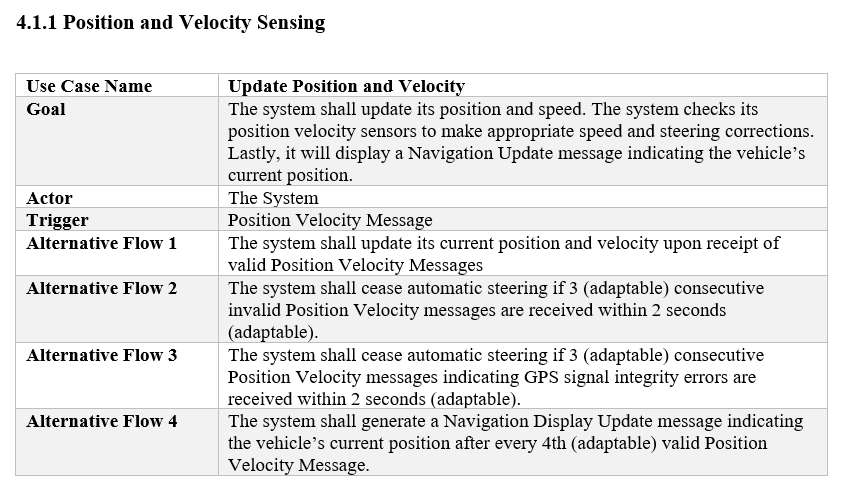
**2. Architectural Representation -**

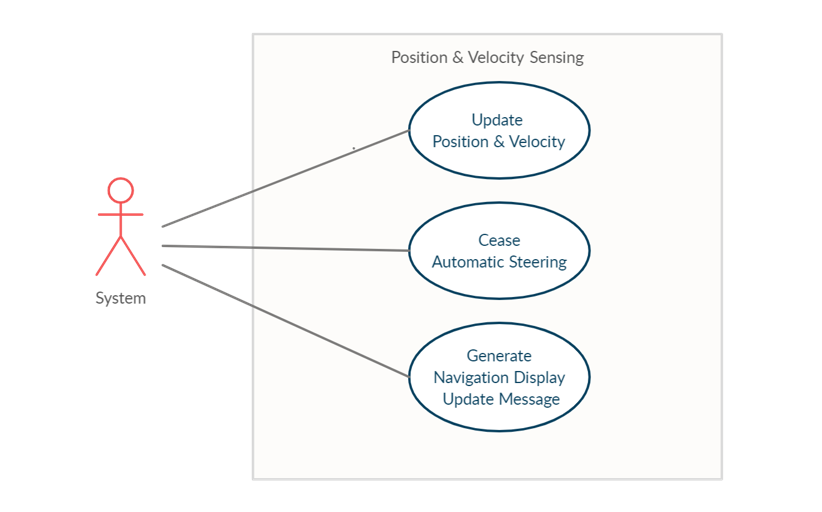
Our software architecture will be represented in this document by the 4+1 View Model. The Use-Case View will describe the functional requirements and the diagrams for each. The Logical view will focus on the functional requirements and the decomposition of the system, including all significant packages and connections between entities. The Process view will describe the lightweight and heavyweight process threads within the system and how they communicate. The Development view will describe the overall structure and the decomposition of the software into layers and subsystems. Lastly, the Physical view will describe how components will be deployed to hardware nodes in the production environment.

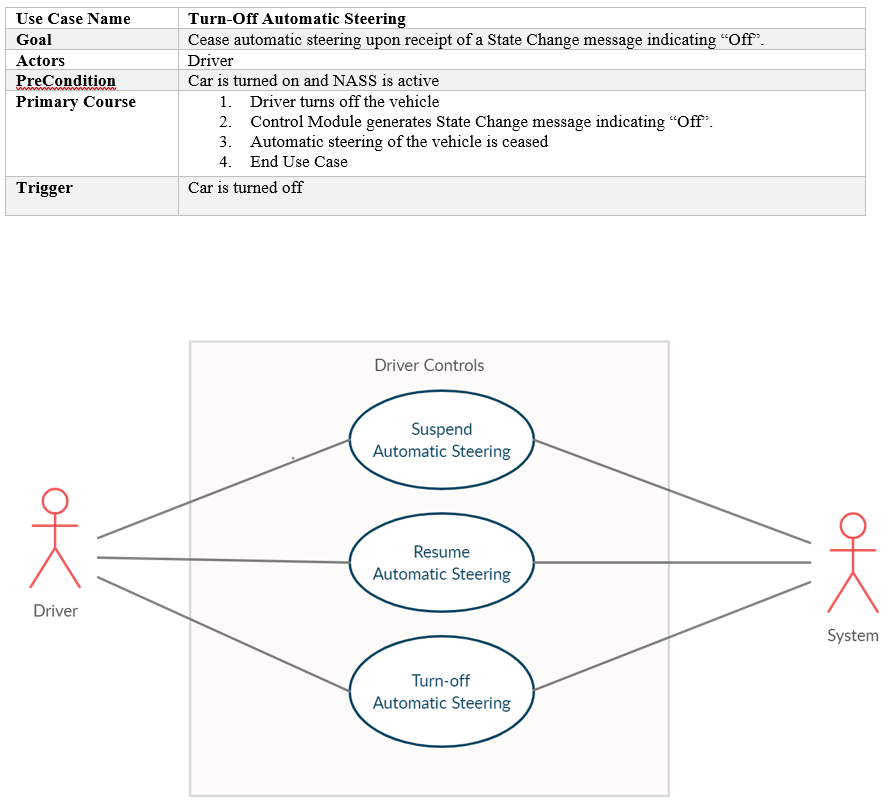
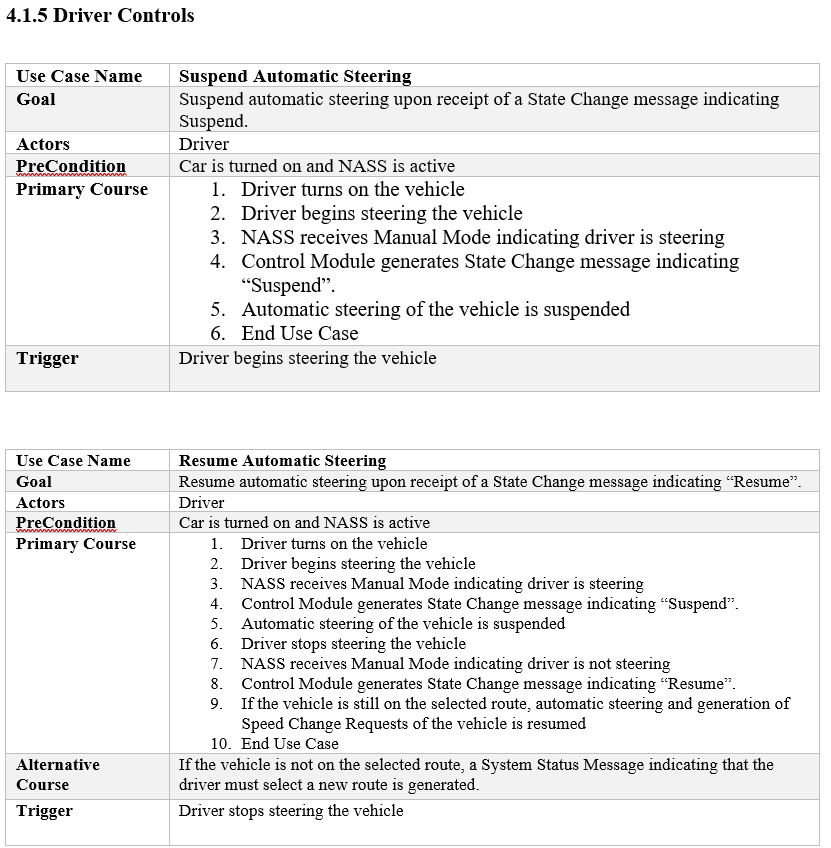
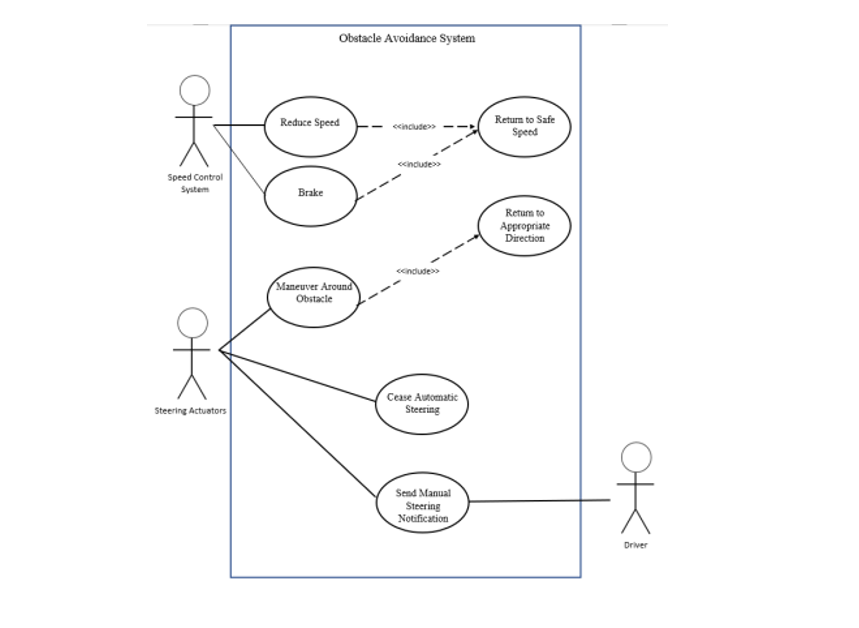
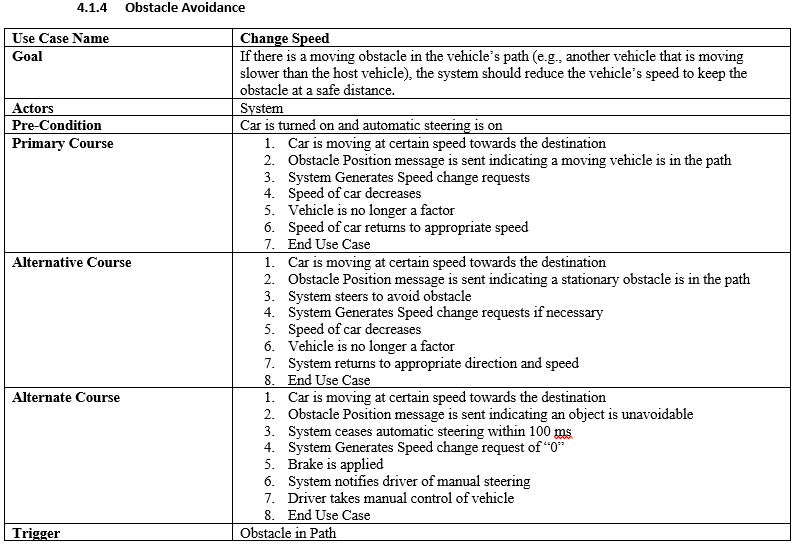
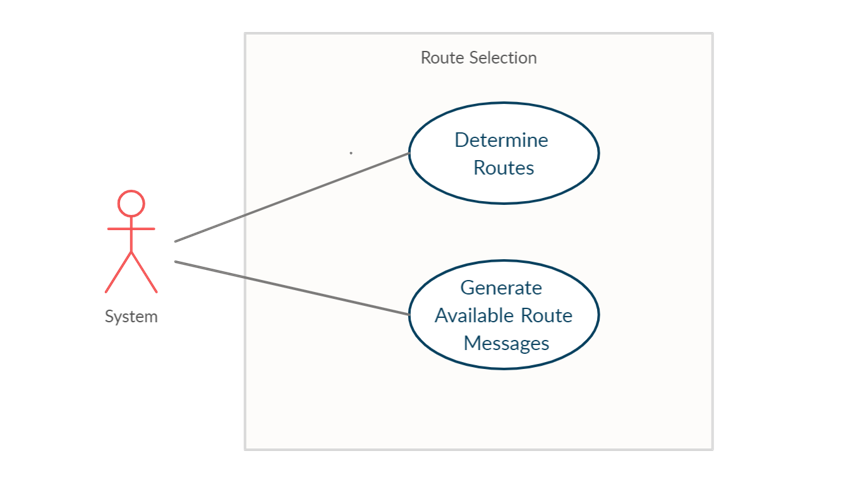
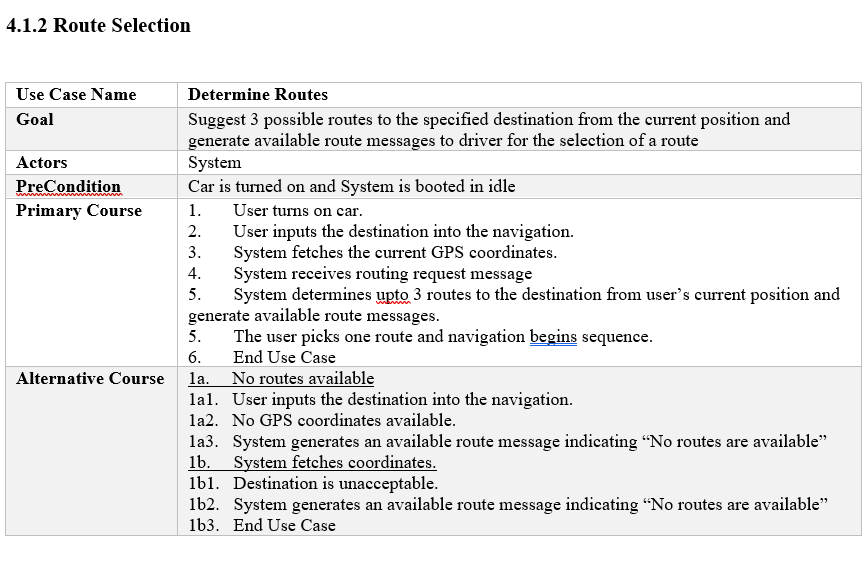
**3. Architectural Goals and Constraints -**

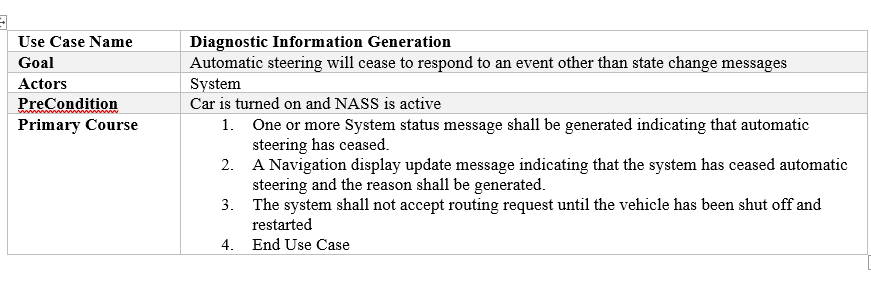
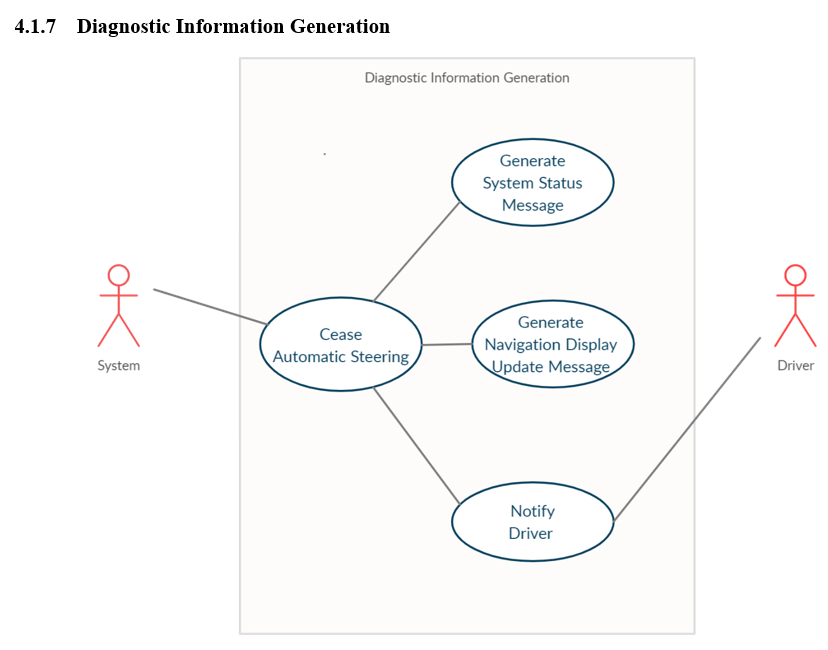
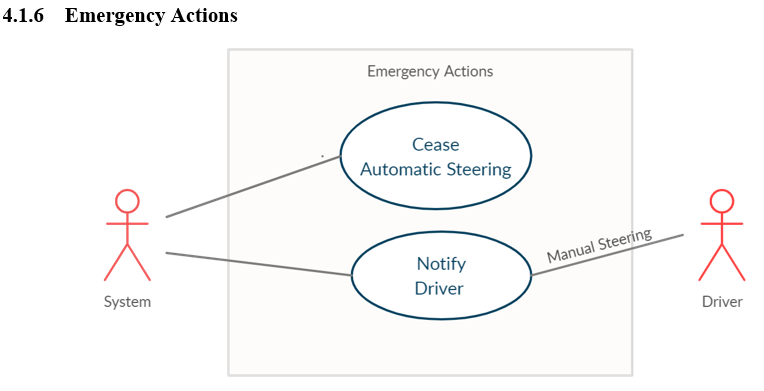
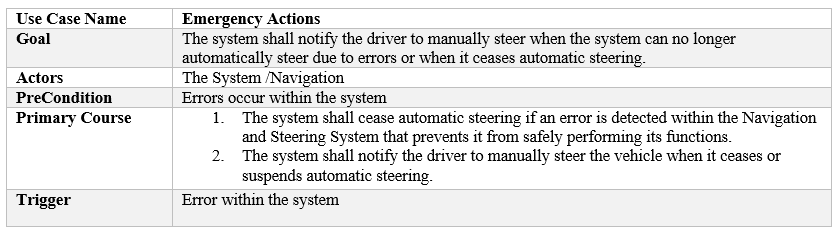
1. The Software shall be optimized for maximum portability. This will ensure that our product can be applied to older models and all vehicles. Also, for updates to be less frequent thus allowing for a longer lasting product.
2. The software must be able to prevent hackers from entering its software and thus will reduce intentional accidents or deaths.
3. The software will aid the driver while in manual mode through audio cues but will not take control of the steering wheel unless the driver allows the software to do so.
4. The software will have an offline saved map of the state it is sold on
5. The software must be user friendly and easy to use for non tech savvy people.
6. All personal data must be encrypted and kept confidential with the permission of the customer.

**4. Use-Case View**

****

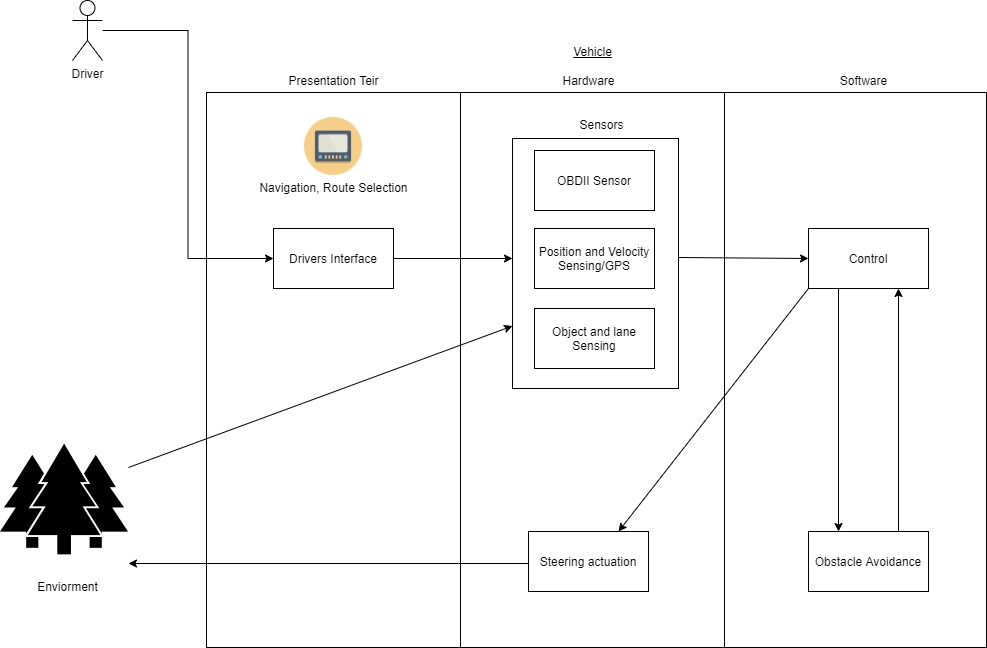
****

****

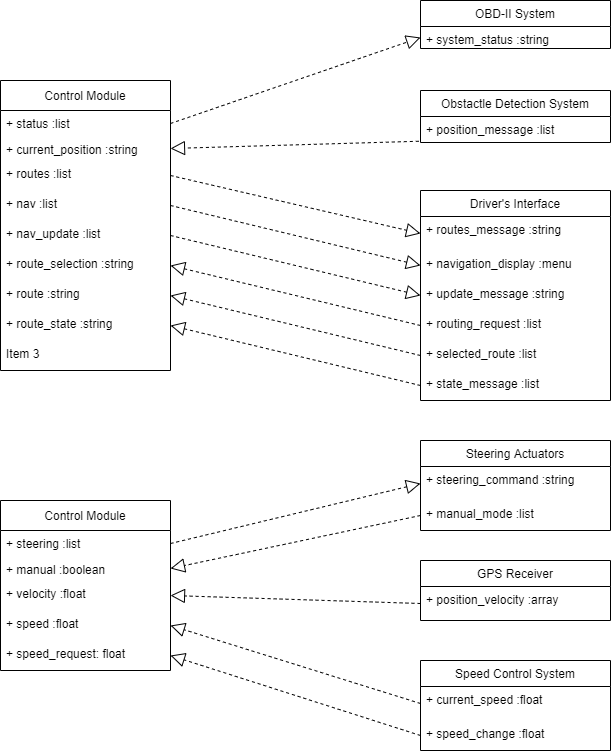
****

**5. Logical View**

This logical view will be a representation of a 4+1 model for our architecture style. The focus will be on the functional requirements and the key abstractions decomposing of the system. This will include conceptual entities such as objects and the connections between them. This view will include diagrams such as class/object diagrams, sequence diagram, state diagram, and activity diagram which will be packaged for a logical view of the product.

****

The Class diagram shows how each major component has significant data types and functions within each class. It also shows how each class is interconnected and the relations each of them have with one another.

**

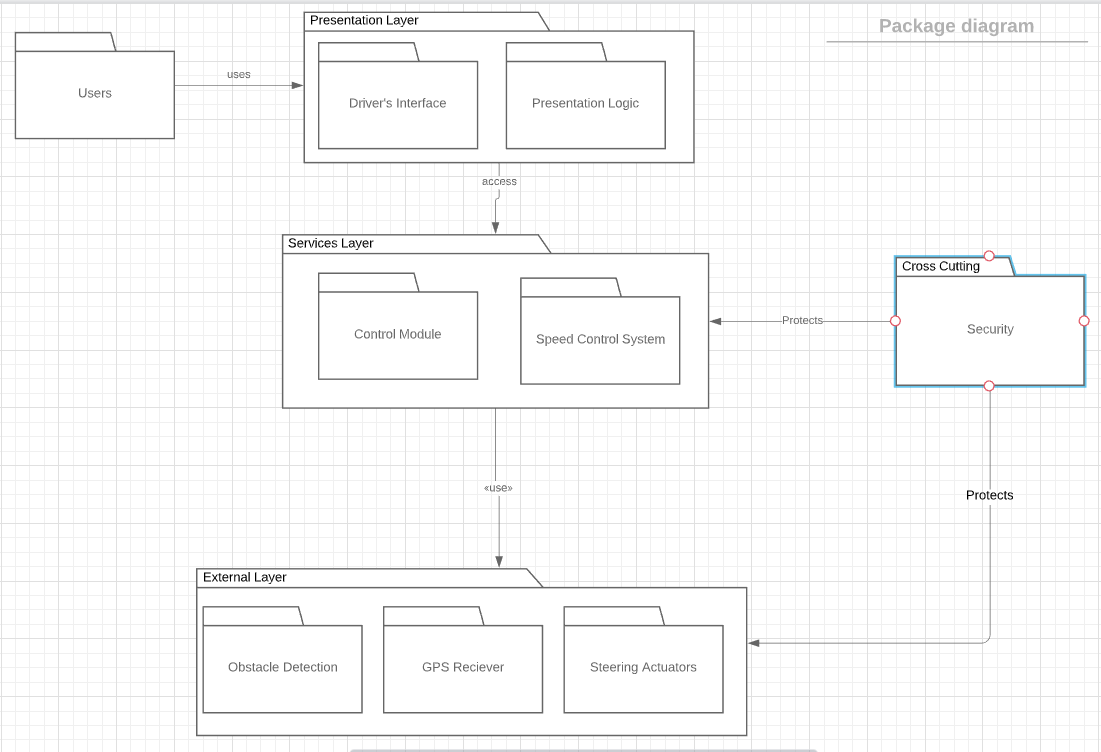
**5.1. Overview**

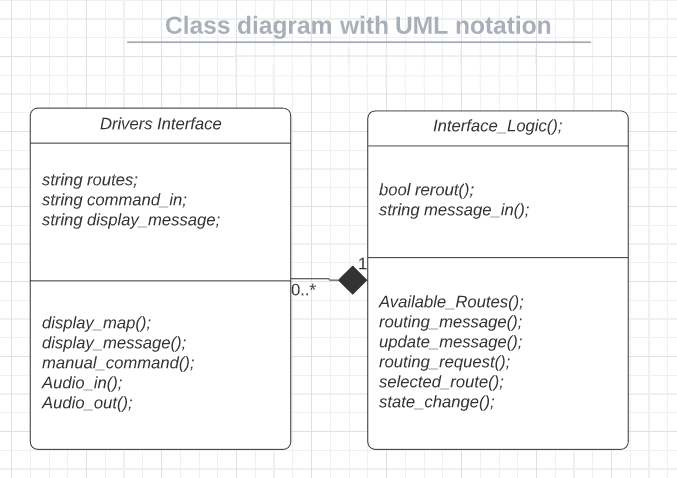
The purpose of the logical view is to specify the functional requirements of the system. The logical view is typically supported by UML static diagrams such as: class/object diagram and UML dynamic diagrams such as interaction overall diagram, sequence diagram, communication diagram, state diagram, and activity diagram

The main artifact of the logical view is the design model:

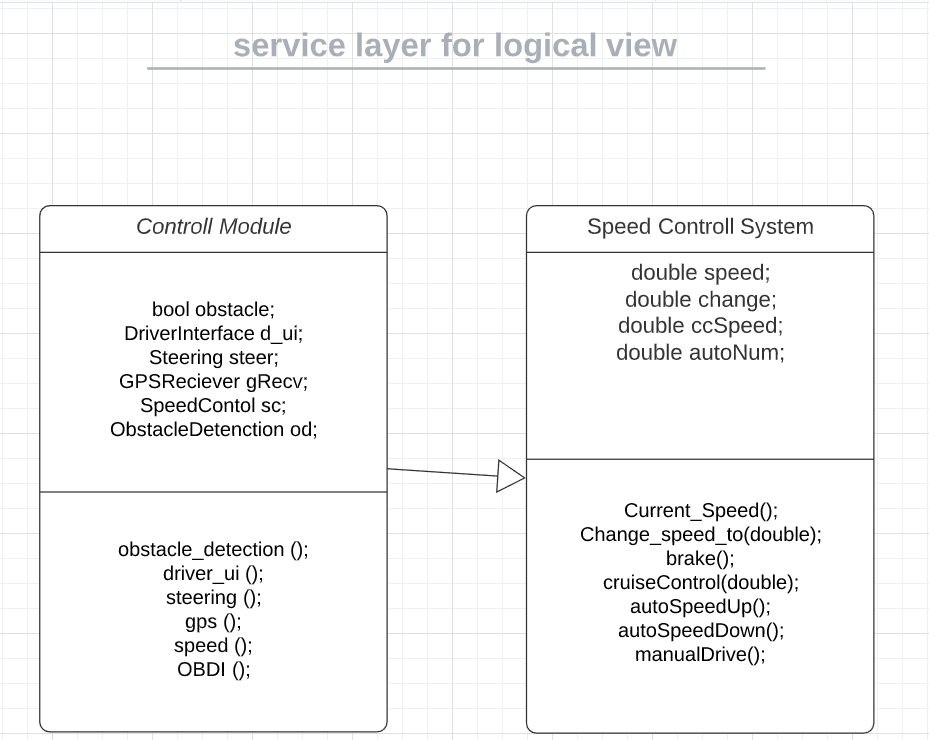
* The design model gives a concrete description of the functional behavior of the system. It is derived from the analysis model.
* The analysis model gives an abstract description of the system behavior based on the use case model.
* In general only the design model is maintained in the logical view, since the analysis model provides a rough sketch, which is later refined into design artifacts.

**5.2. Architecturally Significant Design Packages**

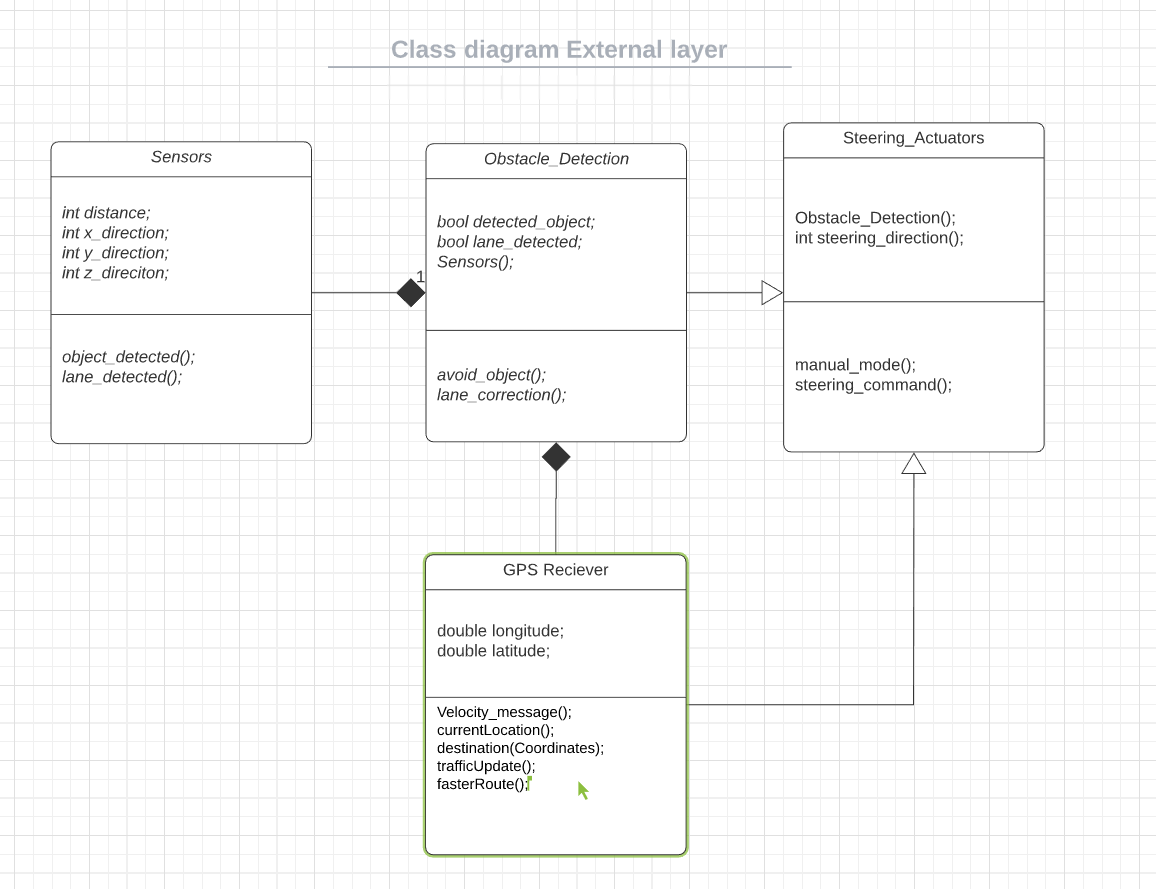
**

**Presentation Layer:** Contains the classes which the user will interact with and the logic needed to correctly respond to users request.****

**Services Layer:** Contains the classes which act between the hardware (external layer) and the user interface (presentation layer)



**External Layer:** Contains the classes which directly interact with hardware(external layer) and then provides that information to the services layer.

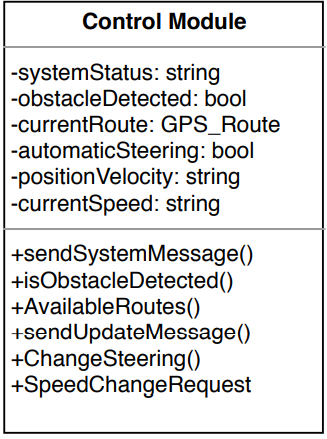


**5.3. Architecturally Significant Design Classes**

**Class: Control Module**

**Description:** Central controller for all subsystems in which the subsystems communicate with.

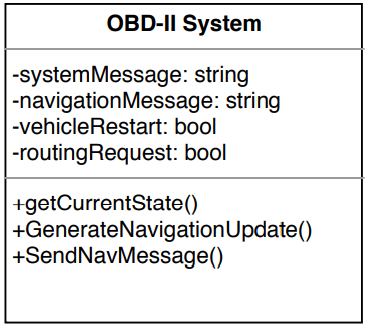
**Diagram:**

****

**Class: OBD-II System**

**Description:** On-Board Diagnostic system that sends System Status messages to display when automatic steering has ceased.

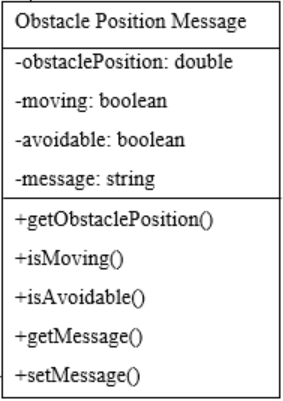
**Diagram:**

****

**Class: Obstacle Detection System**

**Description:** Sends Obstacle Position Message to Control Module for a speed change request to avoid the obstacle if possible, otherwise takes alternative measures.

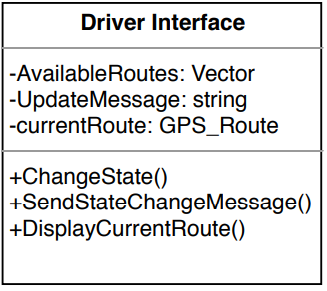
**Diagram:**

****

**Class: Driver Interface**

**Description:** Sends State Change Message to Control Module to indicate that driver wants to suspend automatic steering.

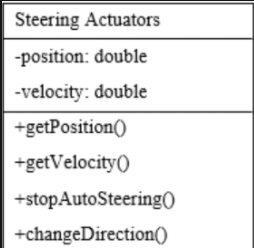
**Diagram:**



**Class: Steering Actuators**

**Description:** Device that uses steering commands from the control module to steer the car and activate/deactivate automatic steering.

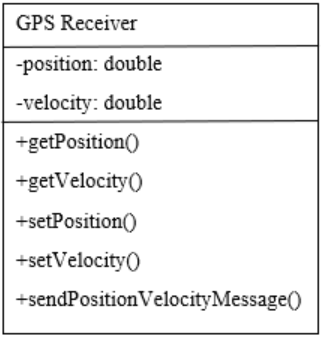
**Diagram:**

****

**Class: GPS Receiver**

**Description:** Device that sends the current position and velocity of the vehicle to the control module.

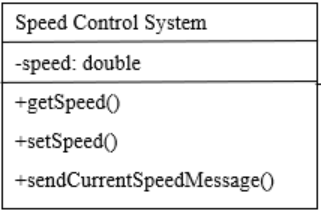
**Diagram:**

****

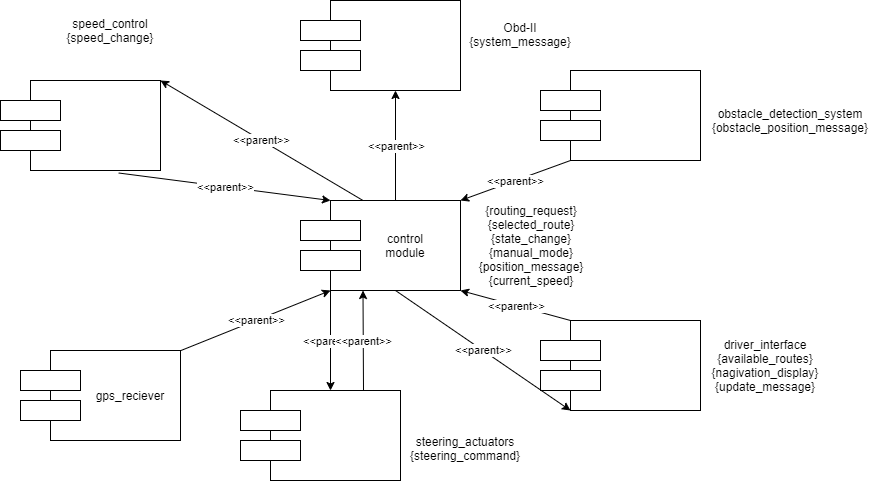
**Class: Speed Control System**

**Description:** System that takes speed change requests, adjusts the vehicles speed, and sends out the current speed to the control module.

**Diagram:**

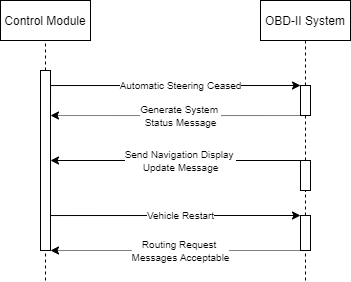
****

**6. Implementation/Development View**

**

**7. Process View**

**OBD-II System Processes**

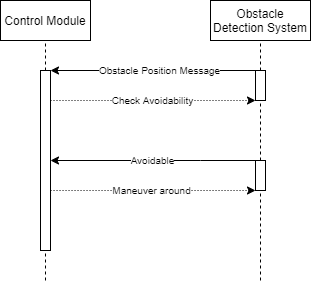
****

Lightweight processes: Automatic Steering Ceased, Generate System Status Message, Send Navigation Display Update Message, Vehicle Restart, Routing Request Message Acceptable

Heavyweight processes: Control Module, OBD-II System

Interprocess Communication: Message Passing

**Obstacle Detection System Processes**

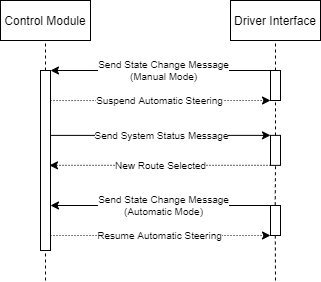
****

Lightweight processes: Obstacle Position Message, Check Avoidability, Maneuver Around

Heavyweight processes: Control Module, Obstacle Detection System

Interprocess Communication: Named Pipes

**Driver’s Interface Processes**

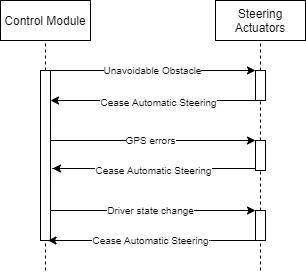
****

Lightweight processes: Send State Change Message, Suspend/Resume Automatic Steering, Send System Status Message, New Route Selected

Heavyweight processes: Control Module, Driver Interface

Interprocess Communication: Message Passing

**Steering Actuators Processes**

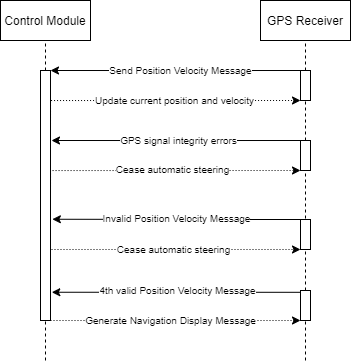
****

Lightweight processes: Check Avoidability, Cease Automatic Steering, Error Handler, Driver State Change

Heavyweight processes: Control Module, Steering Actuators

Interprocess Communication: Shared Memory with Semaphores

**GPS Receiver Processes**

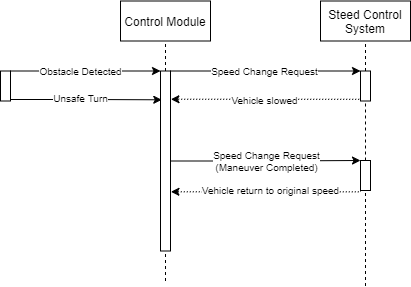
****

Lightweight processes: Send Position Velocity Message, Update Current Position and Velocity, Error Handler, Cease Automatic Steering, Generate Navigation Display Message

Heavyweight processes: Control Module and GPS Receiver

Interprocess Communication: Shared Memory with Semaphores

**Speed Control System Processes**

****

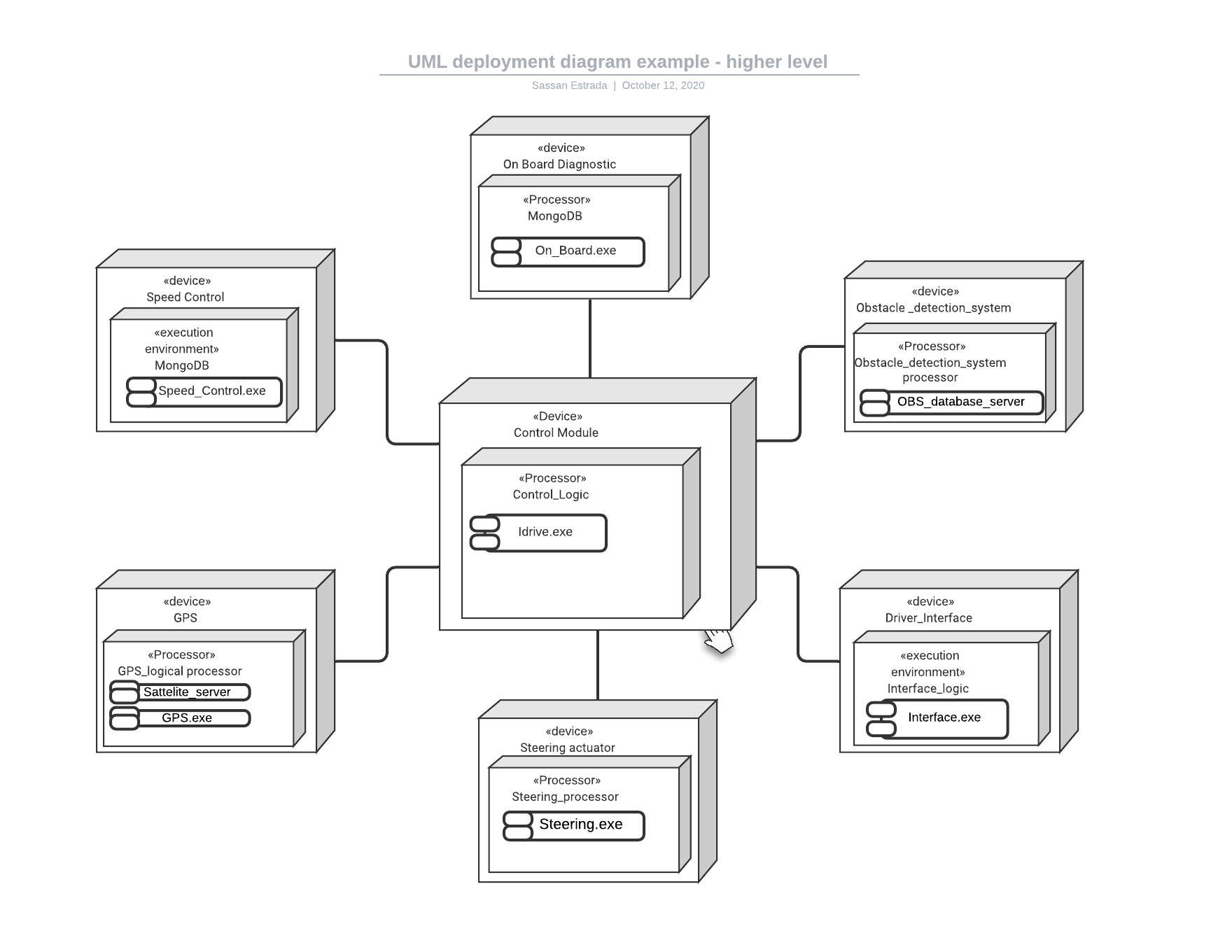
Lightweight processes: Speed Change Request, Obstacle Detected, Change Vehicle’s Speed

Heavyweight processes: Control Module, Speed Control System

Interprocess Communication: Named Pipes

**8. Deployment/Physical View**

The Deployment diagram below shows the control module in the middle representing it is the essential part of the entire system. It is crucial because it connects all the components by collecting all the data and information needed to improve the Idrive system further. The other parts also have significant responsibilities like processors and executables, which help ensure the Idrive system runs as smoothly as possible.

**

**9. Size and Performance -**

iDrive Navigation and Automatic Steering System is scalable to accommodate a multi-year evolution as Sonny Motors Company’s car models change and data loads increase with additional features and improved sensors.

**· Ease to accommodate security features**

The system is designed to prevent unauthorized activity and protect the system from hacking. To provide security, the system will have a touch Id to unlock automatic steering options on the Driver’s Interface.

**· Network failure mechanism to ensure reliability and availability**

When GPS is not available, the system uses the saved offline map to navigate. The system is provided with a Mean Time Between Failures (MTBF) of no less than twelve (12) months. A failure is defined as any error (except the GPS signal integrity error) within the system that causes it to cease automatic steering, or any error that causes the system to behave in an unsafe manner.

**· Accuracy**

The system is able to generate Navigation Display Update messages indicating the vehicle’s current position accurately. The system is able to generate routes for 95% of the test cases to be supplied by the customer.

**· Portability**

The increased use of complementary sensors (such as camera, radar and lidar) shows one reason that electronic systems should be scalable, because scalability easily allows the addition of new sensors to the system. Another reason is that the individual sensors will improve over time, and will require more communications and processing bandwidth. The software is designed to maximize portability. This is to ensure the software can be hosted on vehicles in later model years without substantially redesigning it.

**10. Quality**

For the iDrive system to perform, the following goals have been identified:

**Scalability:**

* **Description:** System’s reaction when many cars are using the same system or like system.
* **Solution:** Each car’s system reacts with GPS and the car’s sensors to handle the location of the system's surroundings.

**Reliability:**

* **Description:** Car’s mechanical system fails.
* **Solution:** OBDII constantly updates to notify the system of any upcoming maintenance, as well as any malfunctions in the system. The system **consists of** a Mean Time Between Failures (MTBF) of no less than twelve (12) months. A failure is defined as any error (except the GPS signal integrity error) within the system that causes it to cease automatic steering, or any error that causes the system to behave in an unsafe manner.

**Availability:**

* **Description:** Car’s navigation system must be connected to the network to provide availability for navigating to destinations at all times.
* **Solution:** The system uses the saved offline map to navigate whenever GPS is unavailable.
* **Description:** The system shall be able to generate routes for 95% of the test cases to be supplied by the customer.

**Portability:**

* **Description:** The system must be designed in such a way that it can be used in any motor vehicle.
* **Solution:** The system is compliant with all major car manufacturers and thus is compliant to their requirements.

**Security:**

* **Description:** System must be secure and inaccessible to non authorized users.
* **Solution:** The system will include state of the art encryption technology.