G-One Debugger

Moway
Software Architecture Document

For Virtual self-driving car using reinforcement learning

Version 1.0

V.Sajeevan
160544C

22th February, 2019

Revision History

Date	Version	Description	Author
22/02/2019	1.0	Document created using initial plans	`V.Sajeeavn

Table of Contents

Software Architecture Document

1. Introduction

The document will give a higher level overview of the architecture of Moway - Virtual self-driving car using reinforcement learning which is a desktop application that can be used by a user who would like to simulate an autonomous vehicle in a virtual environment. Furthermore, different perspectives of the system design will also be illustrated in this document.

1.1. Purpose

The purpose of this software architecture document is to present a detailed architectural description of the Moway, Virtual self-driving car using reinforcement learning. This document will briefly explain the use case view, logical view, process view, implementation view and deployment of the system. This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2. Scope

The scope of this document is to provide a clear description of the architectural development of the system. This will briefly explain the use-case view of the users and system, logical view of the internal system using diagrams, process view, implementation view of the system and deployment view at the final stage. This document is created based on the “4+1” model view of architecture in order to depict the software as accurately as possible.

This document is created based on the “4+1” model view of architecture in order to depict the software as accurately as possible.

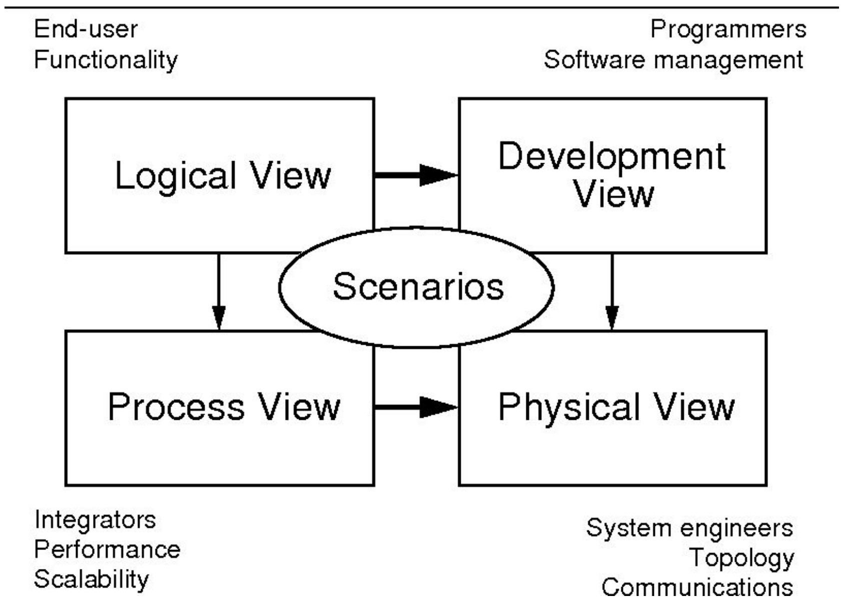

Figure 1 - “4+1” view model

1.3. Definitions, Acronyms, and Abbreviations

Virtual self-driving car	A self-driving car is a vehicle that uses a combination of sensors, and artificial intelligence to travel between destinations without a human operator. Not physically existing as such but made by software to appear to do so.
Rule-based car or manned vehicle	Rule-based cars are used as a way to store and manipulate knowledge to interpret information in a useful way. They are often used in artificial intelligence applications and research
UML	Unified Modeling Language
JIT	Just In Time
SOA	Service Oriented Architecture
RUP	Rational Unified Process
MVC	Model View Controller

1.4. References

[1] Mapping between 4+1 architectural view model & UML.(2019) [online] Available at: https://softwareengineering.stackexchange.com/questions/233257/mapping-between-41architectural-view-model-uml [Accessed 20 Feb. 2019].

The tool used to draw the diagrams:

- Lucidchart [online], www.lucichart.com
- Draw.io [online], www.draw.io

1.5. Overview

The document explains the design of Moway framework and provides a clear understanding of the system. The document is divided into several sections to explain the system clearly

- **Architectural Representation** which describes what software architecture is for the current system, and how it is represented.
- **Architectural Goals and Constraints** which describes the software requirements and objectives that have some significant impact on the architecture.
- **Use-case view** which lists use cases or scenarios from the use-case model.
- The **Logical view** which describes the architecturally significant parts of the design model.
- **Process view** which describes the system's decomposition into lightweight processes.
- **Deployment view** which describes one or more physical network (hardware) configurations on which the software is deployed and run.
- **Implementation view** which describes the overall structure of the implementation model.
- **Size and Performance** which describes the major dimensioning characteristics of the software that impact the architecture.
- The **Quality** which describes the software architecture contributes to all capabilities (other than functionality) of the system.

2. Architectural Representation

The Software Architecture document is developed using the views that are defined in the "4+1" model, but using the RUP (Rational Unified Process) naming convention. The views used to document the system are:

- **Logical view:** It describes the functionality of the system that is provided to the users. It can be described by using a class diagram or sequence diagram.
- **Process view:** It explains the system processes and how they communicate. It can be described by using an activity diagram.
- **Deployment view:** It is related to the topology of software components in the physical layer, as well as the physical connection between these components. It can be described using a deployment diagram.
- **Implementation view:** It is related to software management. It can be described using a component diagram or package diagram.
- **Use case view:** It shows the details of the use cases of the system. Moreover, it describes the services provided by the system to users. It can be described using the use case diagram.

3. Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on architecture.

3.1. Performance

The framework should be user-friendly, appropriate error messages should be displayed and the system should be handling several inputs simultaneously. The system is expected to be responsive all the time with minimum response time.

Since the training through the dataset is computationally exhaustive at times the training gets interrupted and hence it slows down the development of the system. So to mitigate it exploit the different system to develop and simulate.

3.2. Reliability

Accuracy - Automated systems analyzing and forecasting should be very accurate. Forecasting error should be minimum. The forecasted error should be varied within a 5% range from the actual value.

3.3. Security

Security is less priority for this framework. The system will not lose any trained data files from the framework and will make sure that the files are saved in local storage.

3.4. Portability

Since it is a Desktop based application it can be accessed from anywhere, anytime using a Desktop also without internet connection.

3.5. Persistence

Data persistence of the system will be addressed using a database.

3.6. Technical platform

As this is a desktop based application, the users can easily use the system by entering the computer or laptop. They will not require an internet connection to access the framework.

4. Use-case View

The use case view of the system shows all the use cases in the system which represent some of the important functionalities of the system

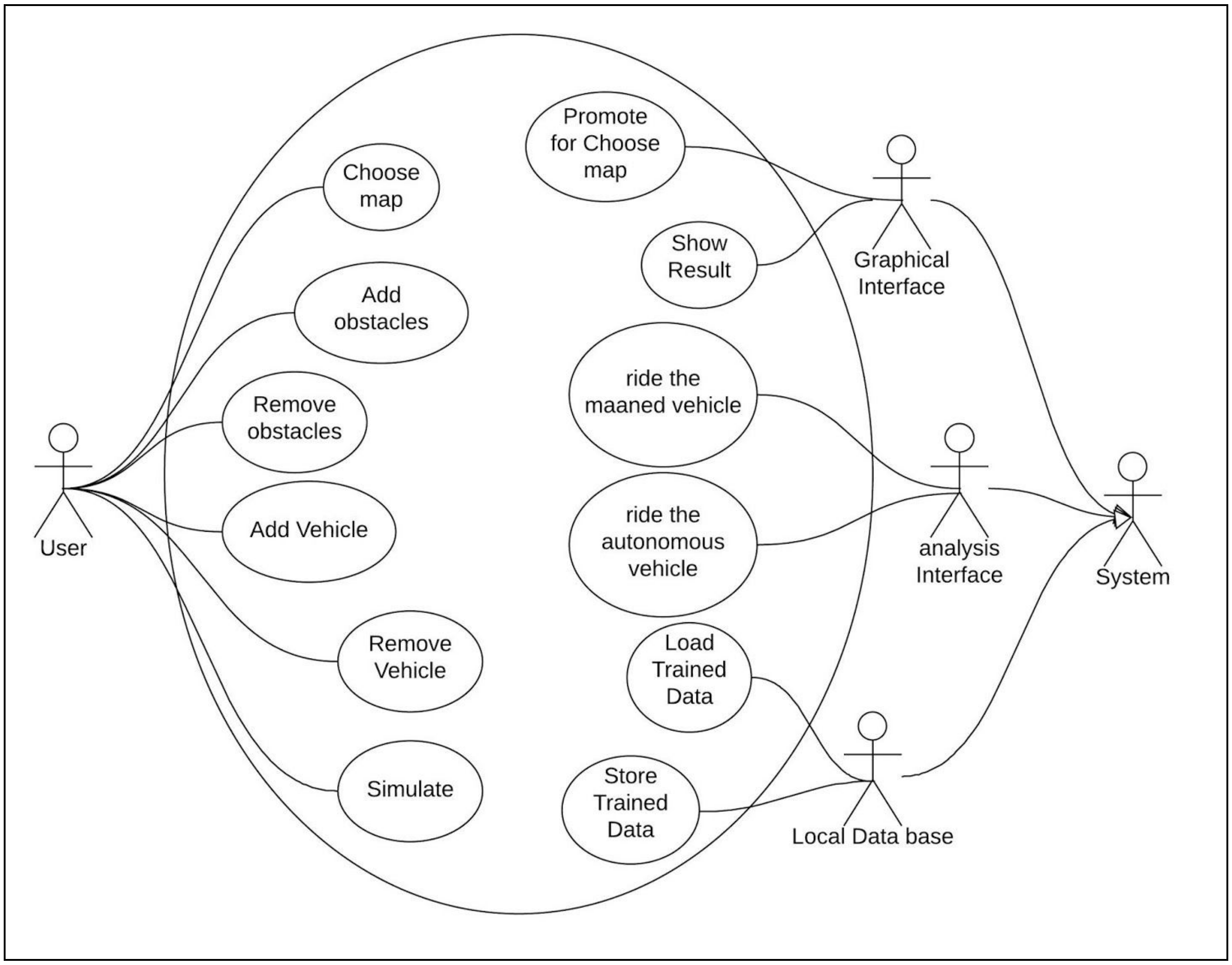

Figure 2- Use Case Diagram

4.1 Use-Case Realizations

The use case realizations will explain how the given use cases actually work in the system. The following use case descriptions will give a clear idea about those functionalities.

- Choose map

Use case name	Choose map for simulate	
Actor	User	
Description	The user should be able to create an environment by choosing different maps.	
Preconditions	-	
Main flow	User	System
	1. View the available map 2. Choose the map	1.Promote available maps
Postcondition	-	
Extensions	-	

- Add obstacle

Use case name	Add obstacle into the map	
Actor	User	
Description	The users want to simulate in a different environment and different phenomena. Such that user can add the obstacles for creating a different environment.	
Preconditions	The user must choose the map	
Main flow	User	System
	1. Select the object 2. Mark the position	1. Check for availability of the object 2. Check for availability of position
Postcondition	Obstacle added into the environment	
Extensions	If the object is not added, notify the user	

- Remove obstacle

Use case name	Remove obstacle into the map	
Actor	User	
Description	The users want to simulate in a different environment and different phenomena. Such that user can remove the obstacles for creating a different environment.	
Preconditions	The user must choose the map	
Main flow	User	System
	1. Select the object 2. confirm	1. Check for availability of the object 2. Promote conform massage
Postcondition	Obstacle removed from the environment	
Extensions	If the object is not removed, notify the user	

- Add Vehicle

Use case name	Add Vehicle into the environment	
Actor	User	
Description	The users want to simulate in a different environment and different phenomena. Such that user can add or remove the Manned vehicles for creating different phenomena.	
Preconditions	The user must choose the map	
Main flow	User	System
	1. Select the vehicle 2. Mark the position	1. Check for availability of the vehicle 2. Check for availability of position
Postcondition	The vehicle added into the environment	
Extensions	If the Vehicle is not added, notify the user	

5. Logical View

The logical view gives an overall representation of the SPMA system without going into detailed architectural levels. The logical view is the object model of the design which concerned with the functionality that the system provides to end-users. UML Diagrams used to represent the logical view include Class diagram, Communication diagram, and Sequence diagram.

The overview of the SPMA system can be shown using a Class diagram as follows.

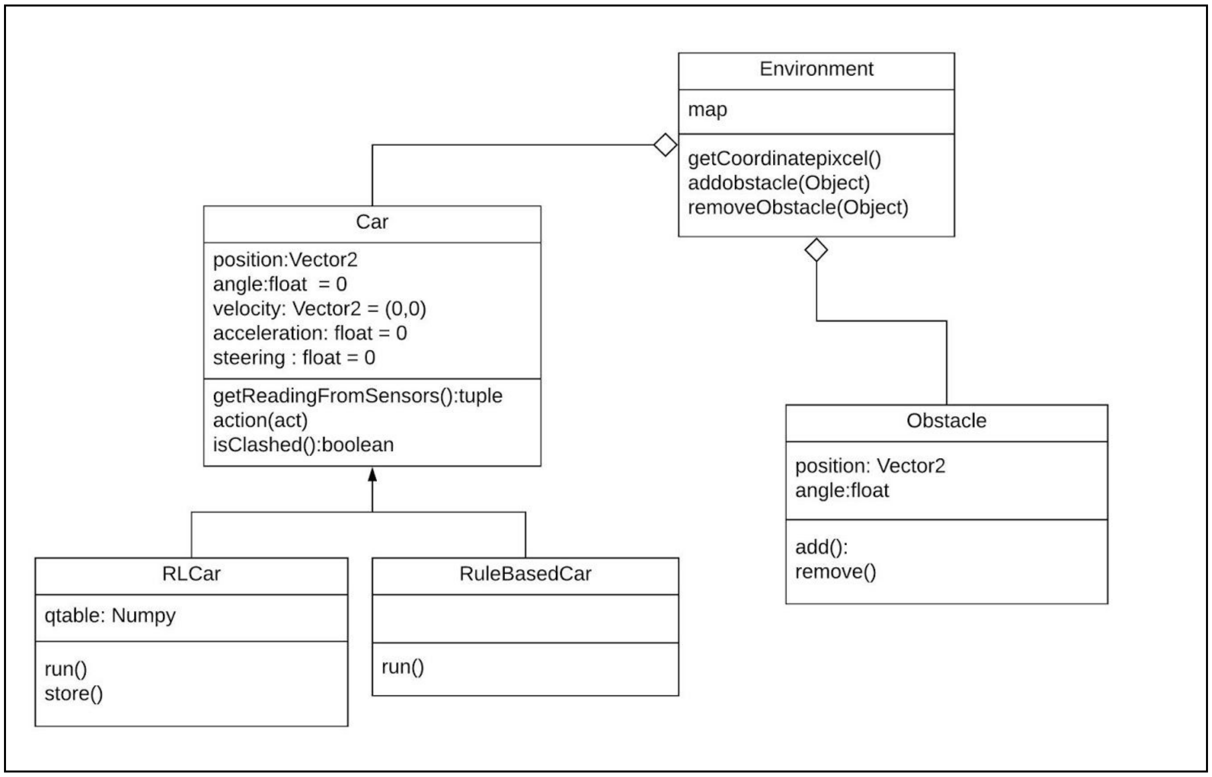

Figure 3- Class Diagram

6. Process View

The process view illustrates how the system behaves at runtime by explaining the system processes and how they communicate dynamically. Following will be the most important processes that have a high impact on the system architecture

6.1 Activity Diagrams

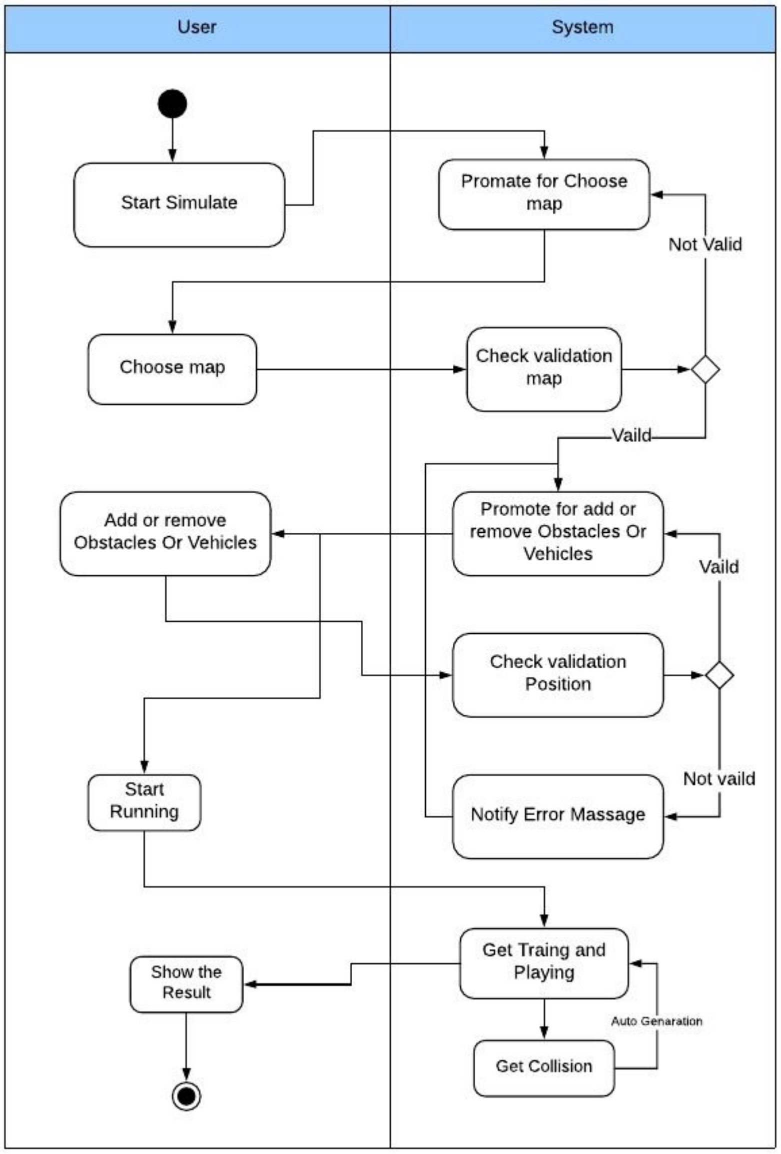

Figure 4 - Activity Diagrams

6.2 Sequence Diagrams

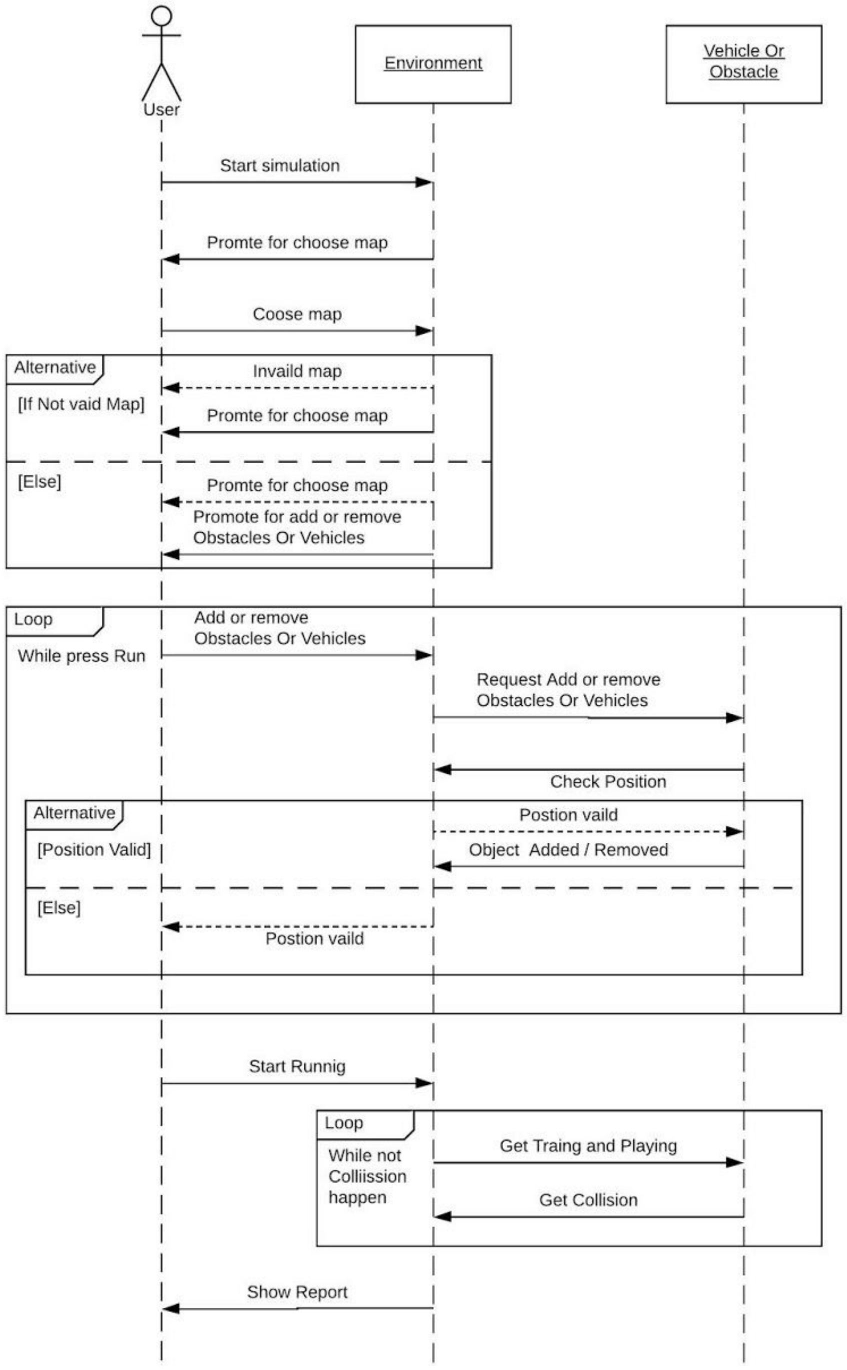

Figure 5 - Sequencel diagram

7. Deployment View

The Deployment view also known as Physical view depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer, as well as the physical connections between these components. UML Diagrams used to represent physical view include the Deployment diagram.

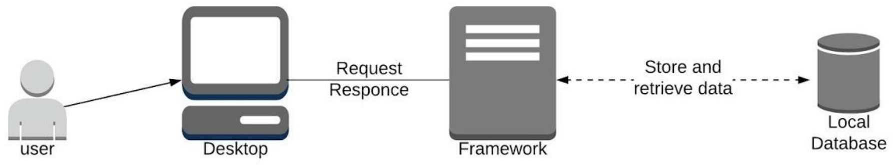

Figure 6 - Deployment Diagram

8. Implementation View

8.1 Overview

For the Moway framework, MVC Architecture is used for Software development. Model – View – Controller Architecture is mostly used in the software industry now.

Models represent knowledge. A model could be a single object or it could be some structure of objects. A view is a representation of its model. It expresses certain attributes of the model and suppresses others. It is acting as a presentation layer

The controller is the link between a user and the system. It provides the user with input by arranging for relevant views to present themselves in appropriate places on the screen. It provides means for user output by presenting the user with menus or other means of giving commands and data. The controller receives such user output, translates it into the appropriate messages and pass these messages on to one or more of the views

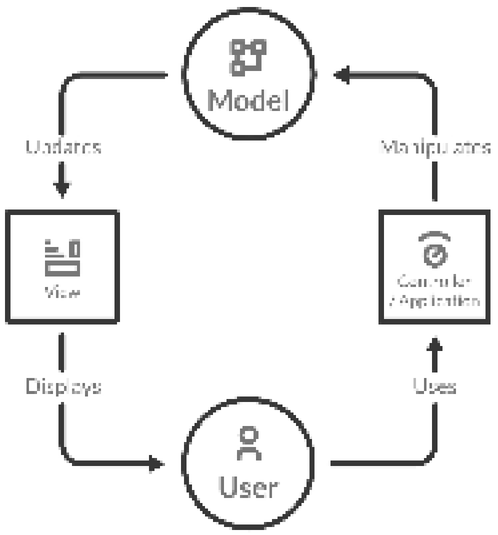

Figure 7- MVC Architecture Diagram

8.2 Layers

Presentation Layer: The presentation layer contains all the front end user components and the UI that are in end-user environments such as simulation UI, Choose map UI. It consists of some logic to provide this framework as well as to validate the input.

Services Layer: The services layer consists of all the services of the system and provides all the business logic of the system. Moreover, it acts as an intermediate link between the presentation layer and the data layer.

Data Layer: The data layer consists of the database. This layer provides persistence for the system and the logic directly related to the manipulation of that data through the means of stored procedures.

9. Size and Performance

Currently, the framework is developed so as to manage about 12 vehicles(manned-vehicle and autonomous vehicle) at a time. It will be expanded to handle a vast number of vehicles in the future

Since the training through the dataset is computationally exhaustive at times the training gets interrupted and hence it slows down the development of the system. So to mitigate it exploit the different system to develop and simulate.

10. Quality

The framework should be user-friendly, appropriate error messages should be displayed and the system should be handling several inputs simultaneously. The system is expected to be responsive all the time with minimum response time.

The user should be easily able to understand the meanings of used UI elements at a glance. The framework should have a clear and flexible structure. Clear and well-formatted coding should be done by following proper coding standards.