Systems design proposal to support the operation of a driverless car

Proposal:

Design an OOP software solution to support the operation of a driverless car using 3 operations inclusive of a user interface. Supports OOP principles including UML and python data search algorithms.

Research: Aims and Objectives.

The research was performed on the current state, evolution, technical aspects and ethical concerns of driverless cars. Search terms were used to understand the concepts of driverless cars to drive the decision-making of the 3 expected operations.

Search terms:

- Autonomous cars
- Autonomous cars testing
- Autonomous cars user expectations
- Pen portrait for driverless car

Objectives:

- To produce UML diagrams to support the design.
- To implement data structures and methods to support the python program.
- To identify 3 operations of a driverless car.

Outcomes:

- Further understanding of the Python language.
- Further understanding of UML and OOP.
- Development of learning and reflective practice.
- Supports testing strategies.
- Further understanding of automation, research and implementation of software in driverless cars.

Design:

In considering the research and the design of software to support the operation of a driverless car, there are common features that should always be present. Metz (2018) illustrates some of the main features of a driverless car which are common across manufacturers. Pen pictures provide a simple way of describing their uses which also supports the selection of the features to be developed. Reddy (2019) discusses the sensors that should be present. From a pragmatic view, it discusses and elaborates on each of the areas. Figure 1, SW Block Diagram (Reddy, 2019) depicts the hierarchical representation of the sensor to the vehicle interface. To provide a simulated program we need to consider components, objects and behaviours from each to develop features. The 6 levels of Vehicle Autonomy Explained. (n.d.). provides a useful hierarchy of automation. This further supports the decision-making in the design. Considering the scope of the assignment it is easy to argue that the deliverables should fall into level 1 or 2 as a semi-automated solution. Also considering the subject areas covered so far in Object Oriented Programming module and the scope of the assignment the following operations have been selected:

Features chosen:

- Sensors (Sensor Class)
- Motor (Drive Class)
- GPS (Journey Class)

    - *User interface (Frontend using the car class as an interface to the three operational classes*

Rationale:

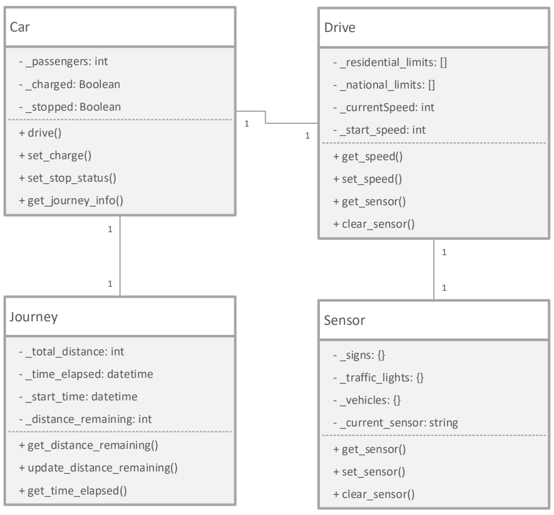
a. Sensors
   A driverless car covers many sensors from GPS to Radar, and LIDAR to Odometers. Sensors cannot just be identified as a single feature as they encompass many technical and system aspects of the car. While considering sensors and simulation of data in a python assignment it would be impossible to cover all these interfaces. Sensors will often work in combination with a LIDAR (Light Detection and Ranging) for the identification and prediction of events. Sensors in this regard will be simulated and use traffic signs, traffic lights and vehicles. These will be classed as dictionaries and lists in Python to provide key pairs and indexed data which will allow logic in class methods to use loops, queues, lists etc to self-assign values and return data to be used in the UI and supporting classes.

b. Motor
   A motor or a mechanism to drive a car is fundamental to a driverless car. By utilising onboard computing, sensors and other features of a driverless car as well as human interaction a motor can be driven or influenced.
   Drive in the context of a class and a motor or actuator provides attributes and methods to control speeds. One of the main aspects of the program will be to combine vehicle speeds with sensors to determine residential or national speeds and based on those attributes and data structures combined with sensors we will be able to determine speeds, reduce speed, and if the car needs to come to a stop. Classes can return data to the UI to provide information.

c. GPS
This will be used to simulate some of the features you would find from a GPS unit. A key component of a driverless car would be to utilise GPS to determine start and end points, traffic alerts, speed limits, and routes. The program will therefore provide a class that can take calculations of current speed and pre-set attributes for the duration to calculate a level of data that will be presented to a UI. By simulating a start and end point or a total distance and by using methods in the classes created we can simulate the values of speed, duration and time to provide a visual interface of human-readable data.

These 3 operations were chosen as they are tangible enough to provide passengers with a real-time understanding of what is happening within the vehicle and of their surroundings. Hein, Rauschnabel, He, Richter & Ivens (2018) discuss some of the risk factors associated with driverless car adoption. By building software that can inform passengers of risks such as speed, warnings, signs etc. then a program that can visualise these as a priority is a must. While some of the risks are identified as a fear of technology if a well-tested and accurate program can improve that confidence then it is beneficial to all.

Diagrams:



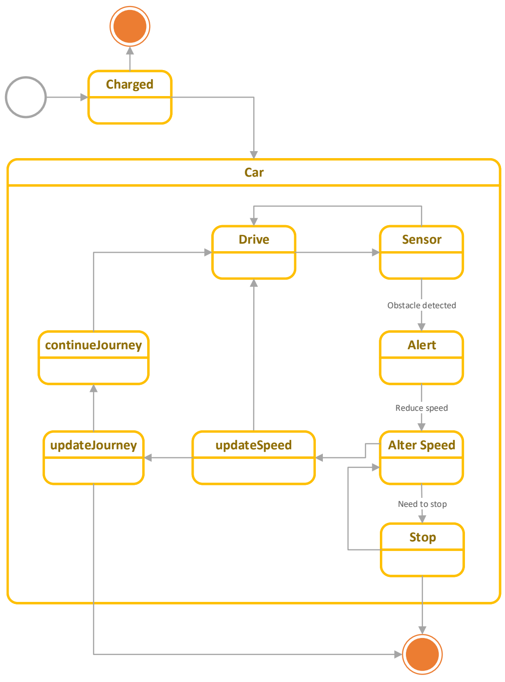


Figure 1: Class Diagram

Figure 2: State Machine Diagram

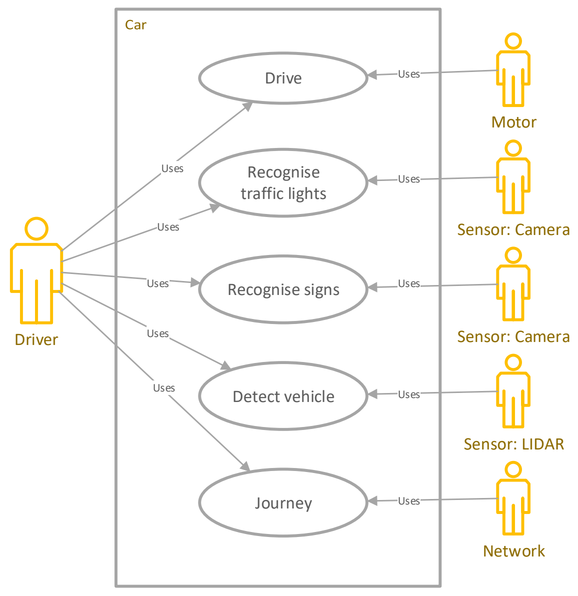


Figure 3: Use Case Diagram

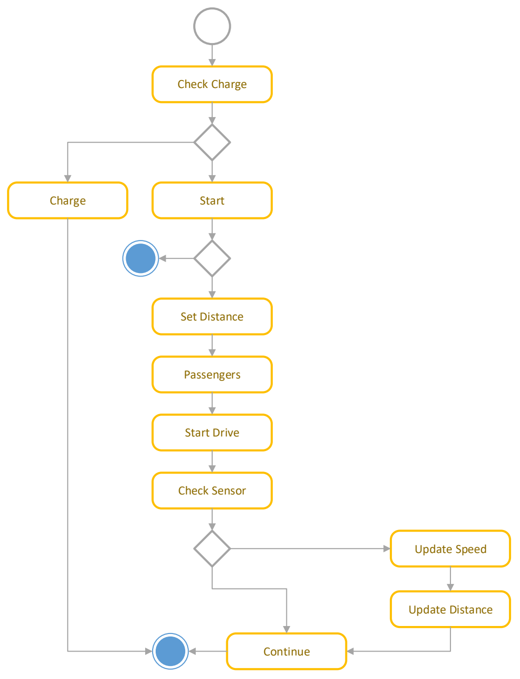


Figure 4: Activity Diagram

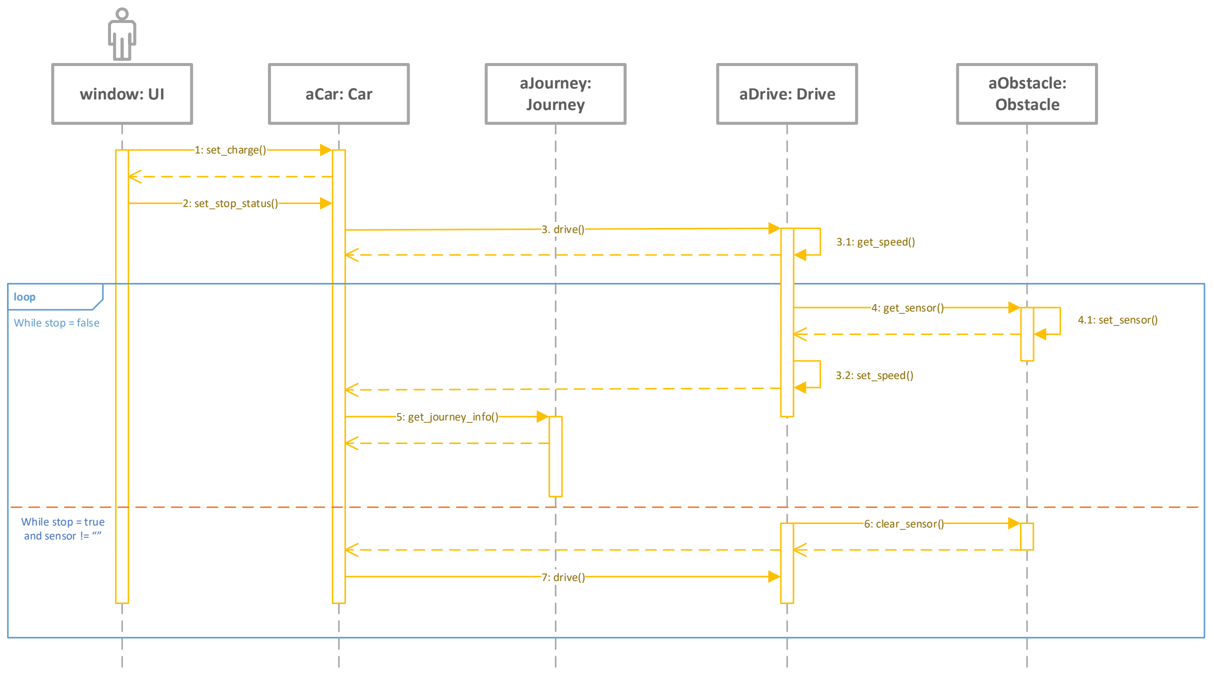


Figure 5: Sequence Diagram

Data Structure:

The following data structures are utilised in the class diagram however as the program is developed it is assumed that these could change based on investigation and development activities.

Type	Example
Dictionary (Use for types with several items like vehicles, signs, lights)	signs = { “sign1”: { “type”: “order”, “name”: “end of 20mph zone” }, “sign2”: { “type”: “warning”, “name”: “Stop 100 yards” },
List (Use for static speed limits).	[0, 5, 10, 20, 30, 40, 50, 70]
Queue (Use for dynamic or simulated values like adding speed from 0 – 70 as vehicle speed is increased simulated)	[~~0~~, 30, 70]
Stack (Use for similar data structure as the vehicle decelerates)	[0, 30, ~~70~~]

References:

Reddy, P. P. (2019) Driverless Car: Software Modelling and Design using Python and TensorFlow.

Zhou, Z. Q. & Sun, L. (2019) Metamorphic testing of driverless cars. *Commun. ACM* 62(3): 61–67. DOI: https://doi.org/10.1145/3241979.

Hein, D., Rauschnabel, P., He, J., Richter, L. & Ivens, B. (2018) What drives the adoption of autonomous cars?

Lin, P. (2016) Why ethics matters for autonomous cars. *Autonomous driving: Technical, legal and social aspects*: 69-85.

The 6 levels of Vehicle Autonomy Explained. (n.d.). Available from https://www.synopsys.com/automotive/autonomous-driving-levels.html [Accessed 9 March 2023].

Metz, C. (2018) How Driverless Cars See the World Around Them. Available from https://www.nytimes.com/2018/03/19/technology/how-driverless-cars-work.html [Accessed 9 March 2023].

Singh, S. & Saini, B.S. (2021) Autonomous cars: Recent developments, challenges, and possible solutions. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1022, No. 1, p. 012028). IOP Publishing.