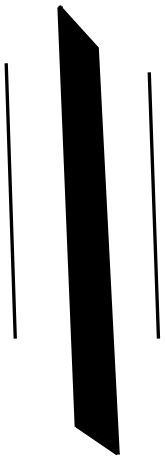
Lamar University

Spring 2013

Software Engineering

PROJECT REPORT ON

“THE HYBRID ENGINE SYSTEM”



Submitted to: Dr. Stefan Andrei

Prepared By:

Guru Acharya(L20302687)

Nischal Budhathoki(L20301126)

Nitin Joshi(L20301507)

Pramit Sinha (L20306876)

ACKOWLEDGEMENT

First and foremost, we would like to express our sincere gratitude to our instructor Dr. Stefan Andrei, professor of Computer science department at Lamar University for providing us the opportunity to work on this project “The Hybrid Engine System” and providing with his valuable and constructive suggestions during the planning and development of this project. It would never have been possible for us to take this project to this level without his innovative ideas, his relentless support and encouragement. His willingness to give his time so generously has been very much appreciated.

We would also like to thank our friends for their constant support, ideas and encouragement throughout the completion of this project.

ABSTRACT

With increasing oil price and mounting environment concerns and increasing number of cars in our modern society, cleaner and sustainable energy solutions have been demanded. Hybrid electrical vehicle is the best solution. In hybrid electric vehicle, it combines an internal combustion engine and electric motor powered by batteries, gives the best features of today's combustion engine cars and electric vehicles. This combination allows the electric motor and a battery to make conventional engine more efficient, which thus reduces the fuel use gives the best mileage.

We design a system that supports the process of simulating a hybrid engine system adopting a use-case driven object-oriented approach. We design a hybrid system which operates in six mode that are Eclectic Vehicle mode, Cruise mode, Overdrive mode, Battery charge mode, Power boost mode and Negative split mode.

Language used: java

Platform: JDK 1.6, SDK

Editor: Eclipse Juno

1 Table of Contents

Table of Figures:

1. Introduction

A Hybrid engine is a type of engine which combines a conventional Internal Combustion Engine (ICE) propulsion system with an electric propulsion system. Due to the presence of the electric power-train it achieves better fuel economy and better performance than the traditional conventional vehicle. Hybrid engines are smaller and more efficient than traditional fuel engines. Some hybrid vehicles use regenerative braking to generate while travelling. There is a variety of Hybrid Engine Vehicle (HEV) but the most common form of HEV is the hybrid electric car.

1.1. Hybrid System Engine Structure:

In hybrid electric drive there are two motors: an eclectic motor and an internal combustion engine. The planetary gear set (power splitter) is used for sharing the power from electric motor and an internal combustion engine. The ratio can be from 0-100% for the combustion engine, or 0-100% for the electric motor, or anything in between, such as 40% for the electric motor and 60% for the combustion engine. The following figure shows the structure of the hybrid engine vehicle.

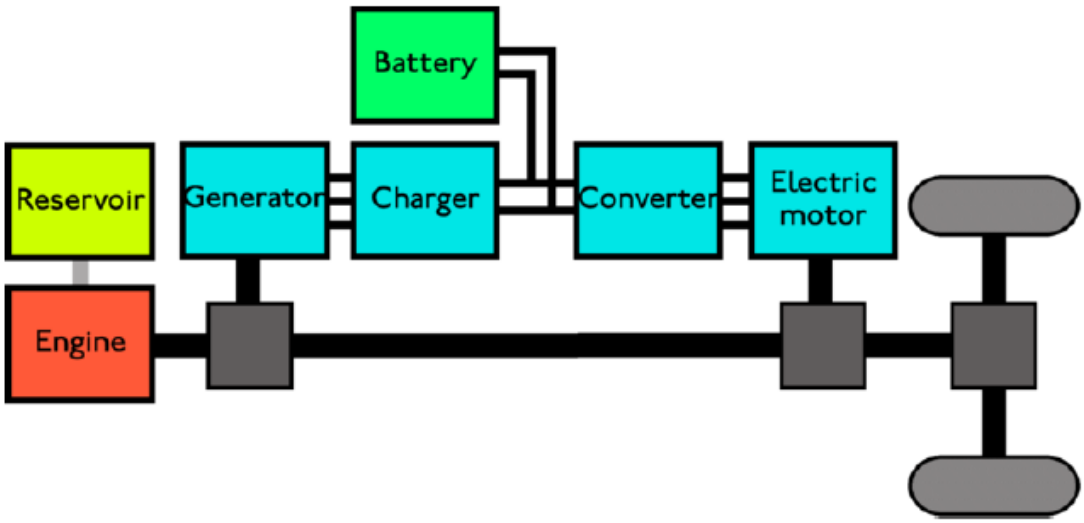


Figure 1-1Hybrid engine system

The above figure shows the structure of the hybrid engine system where electric motor can act as a generator for charging the batteries. A full hybrid is a vehicle which can run on just the engine, just the batteries, or a combination of both. For full hybrid vehicle, a large high-capacity battery pack is needed for battery –only operation. . These vehicles have a split power path that allows more flexibility in the drive train by inter-converting mechanical and electrical power, at some cost in complexity. To balance the forces from each portion, the vehicles use a differential-style linkage between the engine and motor connected to the head end of the transmission. Toyota Prius, Toyota Camry Hybrid, Ford Escape Hybrid/Mercury Mariner Hybrid, Ford Fusion Hybrid/Lincoln and many more are the example of full hybrid vehicle.

The Toyota has brand name for this technology is Hybrid Synergy Drive, which is being used in the Toyota Prius, the Highlander Hybrid SUV, and the Camry Hybrid. The operation of the Prius can be divided into six distinct regimes or modes:

1**. Electric vehicle mode**: The engine is off, and the battery provides electrical energy to power the motor (or the reverse when regenerative braking is engaged). Used for idling as well when the battery State Of Charge (SOC) is high.

2. **Cruise mode:** The vehicle is cruising (i.e. not accelerating), and the engine can meet the road load demand. The power from the engine is split between the mechanical path and the generator. The battery provides electrical energy to power the motor, whose power is summed mechanically with the engine. If the battery state-of-charge is low, part of the power from the generator is directed towards charging the battery.

3. Overdrive mode: A portion of the rotational energy is siphoned off by the main electric motor, operating as a generator, to produce electricity. This electrical energy is used to drive the sun gear in the direction opposite its usual rotation. The end result has the ring gear rotating faster than the engine, albeit at lower torque.

4. Battery charge mode: Also used for idling, except that in this case the battery state-of-charge is low and requires charging, which is provided by the engine and generator?

5. Power boost mode: Employed in situations where the engine cannot meet the road load demand. The battery is then used to power the motor to provide a boost to the engine power.

6. Negative split mode: The vehicle is cruising and the battery state-of-charge is high. The battery provides power to both the motor (to provide mechanical power) and to the generator. The generator converts this to mechanical energy that it directs towards the engine shaft, slowing it down (although not altering its torque output). The purpose of this engine "lugging" is to increase the fuel economy of the vehicle.

Hybrid vehicles can be used in different modes which are shown in below figure:

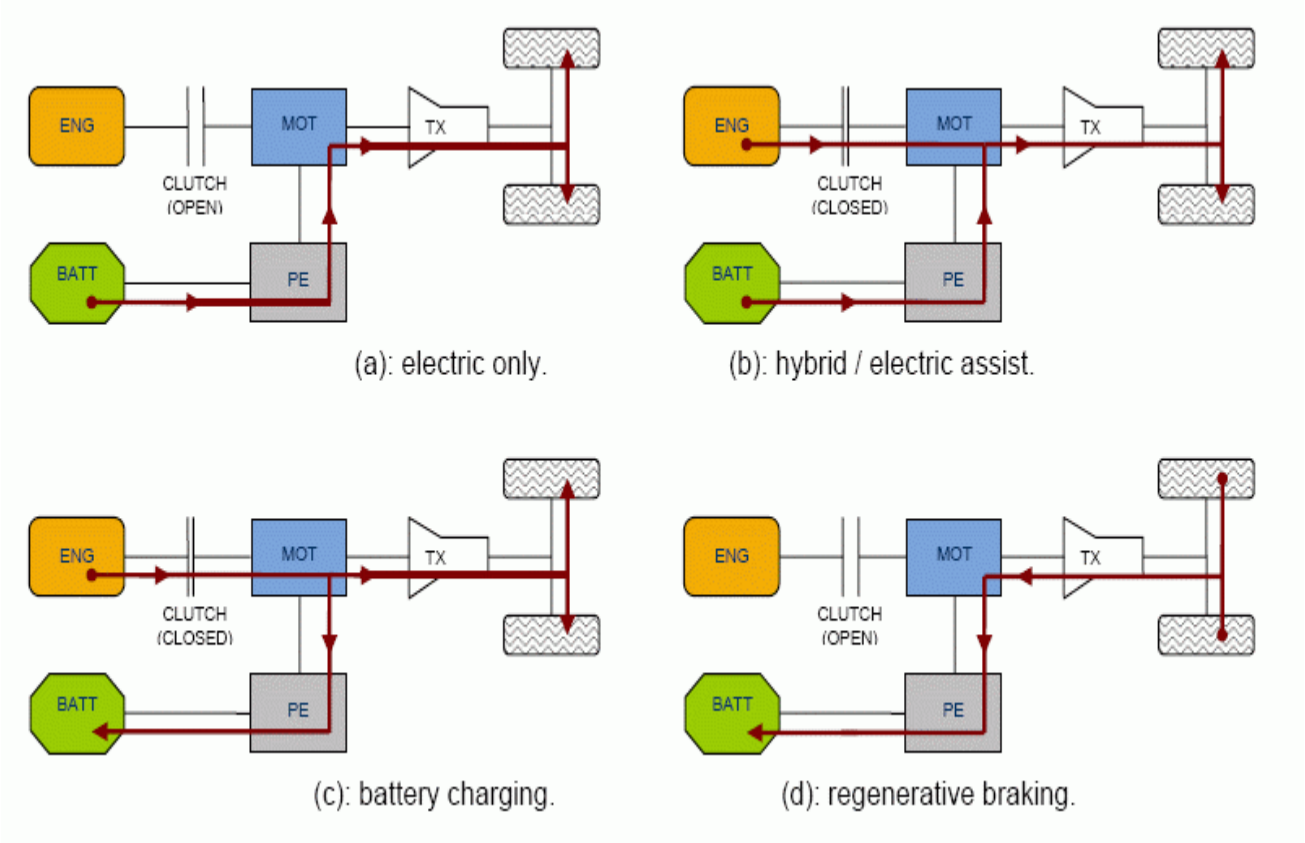


Figure 1-2 different modes in hybrid engine

2 Hybrid Engine System Simulation model

Our program simulate the gas consumption of a trip from point A to B for a given hybrid engine vehicle, a number of gallons for gas and weight added to the vehicle. This system also has a map we use GPS which gives the information about the road condition between point A and point B, including the number of miles, information about the ramps and slopes, speeds and more. This system also follow the speed limit and also shows the how much efficient of the system.

2.1 Use Case Diagram

A **use case** in software engineering and systems engineering is a description of a system's behavior as it responds to a request that originates from outside of that system. In other words, a use case describes "who" can do "what" with the system in question. The use case technique is used to capture a system's behavioral requirements by detailing scenario-driven threads through the functional requirements. Use cases describe the system from the user's point of view. Within systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. Each use case focuses on describing how to achieve a goal or a task. For most software projects, this means that multiple, perhaps dozens of use cases are needed to define the scope of the new system. The degree of formality of a particular software project and the stage of the project will influence the level of detail required in each use case.

A use case may be related to one or more features, and a feature may be related to one or more use cases. A use case defines the interactions between external actors and the system under consideration to accomplish a goal. An actor specifies a role played by a person or thing when interacting with the system. The same person using the system may be represented as different actors because they are playing different roles. Use cases treat the system as a black box, and the interactions with the system, including system responses, are perceived as from outside the system. This is a deliberate policy, because it forces the author to focus on what the system must do, not how it is to be done, and avoids the trap of making assumptions about how the functionality will be accomplished. Use cases may be described at the abstract level (business

use case, sometimes called essential use case), or at the system level (system use case).use case involves

Basic course of events*:* At a minimum, each use case should convey a *primary scenario*, or typical course of events, also called "basic flow", "normal flow," "happy flow" and “happy path". The main basic course of events is often conveyed as a set of usually numbered steps.

Alternative paths or Extensions: Use cases may contain secondary paths or alternative scenarios, which are variations on the main theme. Each tested rule may lead to an alternative path and when there are many rules the permutation of paths increases rapidly, which can lead to very complex documents. Sometimes it is better to use conditional logic or activity diagrams to describe use case with many rules and conditions.

Exceptions, or what happens when things go wrong at the system level, may also be described, not using the alternative paths section but in a section of their own. Alternative paths make use of the numbering of the basic course of events to show at which point they differ from the basic scenario, and, if appropriate, where they rejoin. The intention is to avoid repeating information unnecessarily. The description of an exception should indicate how the system will respond to, or (if possible) recover from, the error condition.

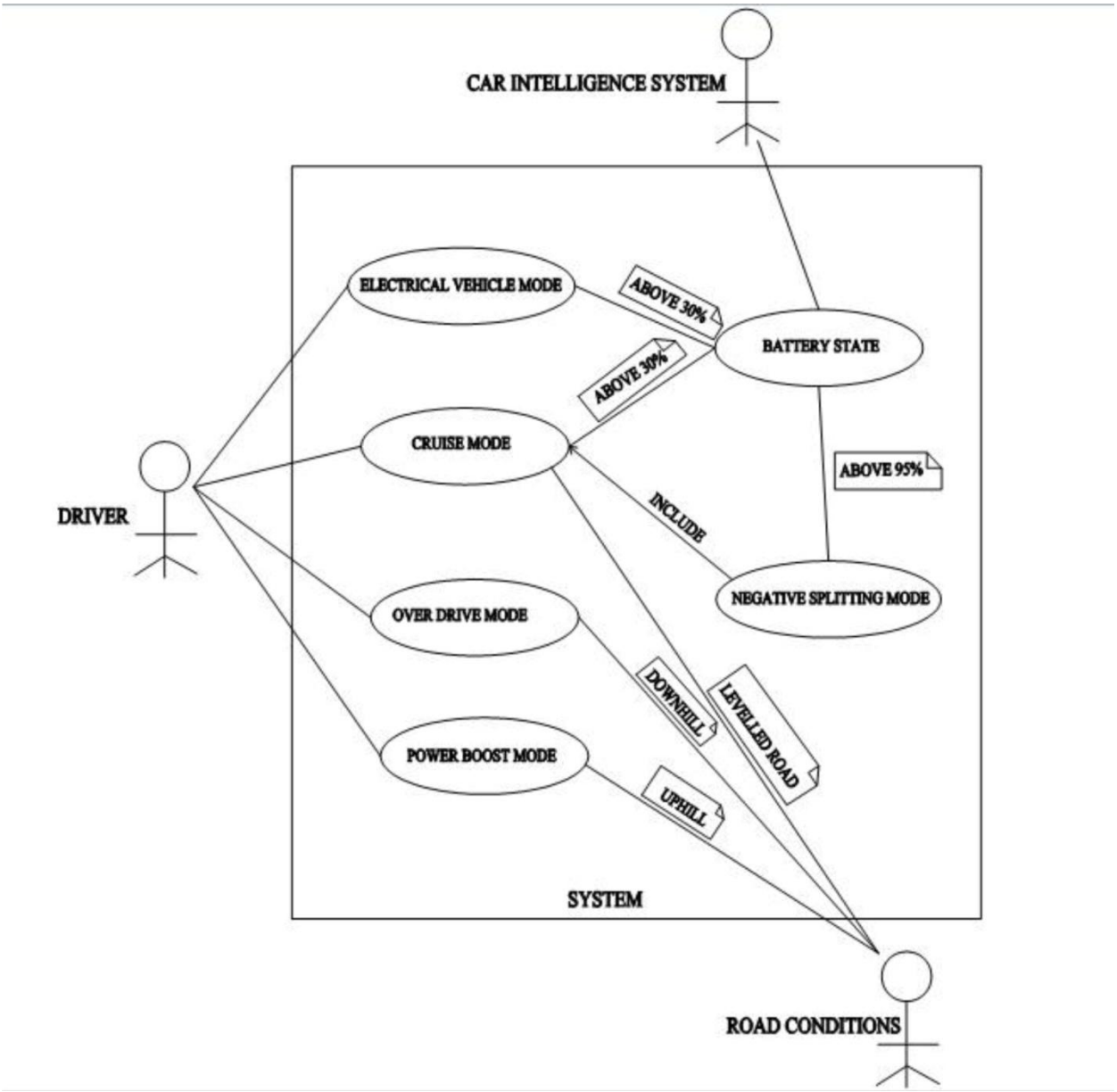


Figure 2-1 Use case diagram of hybrid engine

ELECTRICAL VEHICLE MODE:

Basic course of Events:

1. Driver presses electrical vehicle mode button.
2. Engine is off and battery is used for running the engine.

Exceptional course of Event:

1. Driver presses electrical vehicle mode button.
2. Battery charge is below 30 % and vehicle is run by engine.

CRUISE MODE:

Basic course of Events:

1. Driver presses cruise mode button.
2. The engine can meet the road load demand. The power from the engine is split between the mechanical path and the generator.

Exceptional Course of Event:

1. Driver presses cruise mode button.
2. Road is not plane and it goes to Overdrive mode or power boost mode depending upon the condition of road.

OVER DRIVE MODE:

Basic course of Events:

1. Driver sees downhill road and presses Over Drive Mode button.
2. A portion of the rotational energy is siphoned off by the main electric motor, operating as a generator, to produce electricity. This electrical energy is used to drive the sun gear in the direction opposite its usual rotation.

Exception Course of Events:

1. Driver presses Over Drive Mode button.
2. Vehicle is way slower than speed limit. Vehicle goes to electric vehicle mode

POWER BOOST MODE

Basic Course of events:

1. Driver sees uphill road and presses Power Boost Mode.

2. The engine cannot meet the road load demand then battery is used to power the motor to provide a boost to the engine power.

Exceptional Course of events:

1. Driver presses the Power Boost Mode.
2. The battery charge is not sufficient then only engine is used to power the motor.

NEGATIVE SPLITING MODE

Basic course of events:

1. Driver presses Negative Splitting Mode
2. The vehicle is cruising and the battery state-of-charge is high. The battery provides power to both the motor and to the generator. The generator converts this to mechanical energy that it directs towards the engine shaft, slowing it down. The purpose of this engine "lugging" is to increase the fuel economy of the vehicle.

Exceptional Course of events:

1. Driver presses Negative Splitting Mode.
2. Battery charge state is low and only engine drives the vehicle.

3 Domain Model

A domain model is also known as an Domain Object Model and in software engineering terms it’s a conceptual model which describes the various entities involves in the model and the relationships involved between them. Documentation of the key concepts is the main purpose of the Domain Model. The model identifies the relationships among all major entities within the system, and usually identifies their important methods and attributes. The domain model provides a structural view of the system that can be complemented by other dynamic views in Use Case models. It describes and constraints of the system scope and can be effectively used to

verify and validate the understanding of the problem domain among various stakeholders of the project group. It is especially helpful as a communication tool and a focusing point.

3.1 Description of Domain Model:

The most common way to document the business concepts use in use cases description is to present the domain model. Here domain model should identify concepts between driver and traffic law is first driver look at signal then watch out for traffic and pedestrian and the driver do his action. Concepts between traffic light and traffic light activities are, for the traffic light phase, cycle and time settings are set and traffic light work as per that.

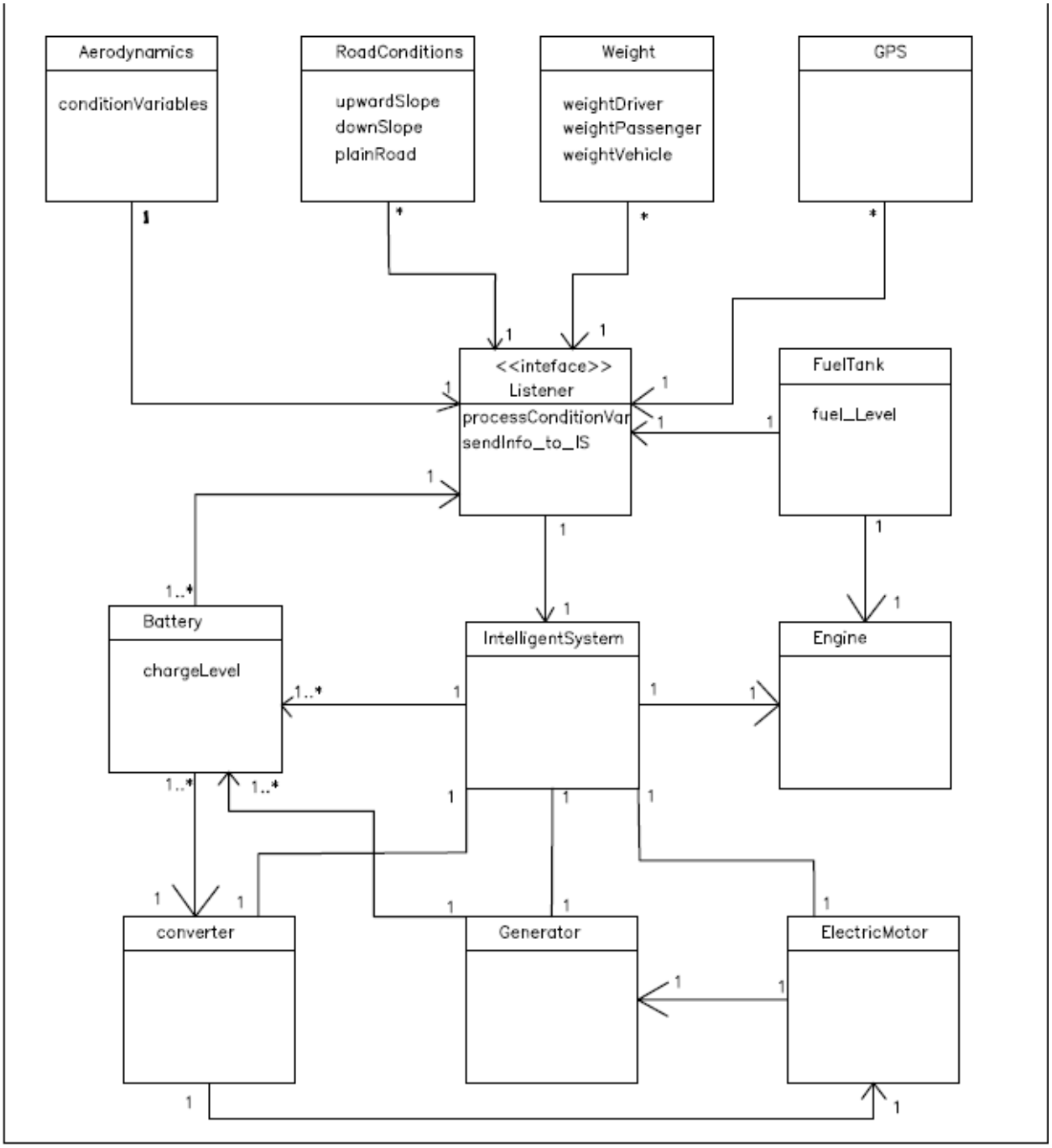


Figure 3-1 Domain model of hybrid engine

4 Sequence Diagram:

Sequence diagrams are used to create one scenario of events through a use case. The objects associated with this series of events and the interactions between the objects are identified. These interactions are characterized by messages sent between the objects, but sequence diagrams do describe how the objects are linked. Objects in a sequence diagram are arranged in columns on the horizontal axis of the diagram. Each object is represented with the object name inside of a rectangle. The object's label takes the form of ClassName:objectName. An external object that interacts with the system is represented by a stick figure. A vertical line is dropped down from each object. This vertical line represents the lifeline of the object in the system. Time is represented by the vertical axis of the diagram. A message is represented by an arrow that points to the receiver. An object may send a message to itself. The message is labeled with the name of the message that is being passed, and any other arguments associated with the message.

As parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

If the lifeline is that of an object, it demonstrates a role.Note that leaving the instance name blank can represent anonymous and unnamed instances. In order to display interaction, messages are used. These are horizontal arrows with the message name written above them. Solid arrows with full heads are synchronous calls, solid arrows with stick heads are asynchronous calls and dashed arrows with stick heads are return messages. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message. Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing.When an object is destroyed (removed from memory), an X is drawn on top of the lifeline, and the dashed line ceases to be drawn below it. It should be the result of a message, either from the object itself, or another. A message sent from outside the diagram can be represented by a message originating from a filled-in circle or from a border of sequence diagram.

4.1 Description of Sequence diagram:

In sequence diagram time passes from top to bottom. In this Traffic signal system diagram classes and actors are at the top. Here messages are shown as arrow between lifelines. Here each operation is labeled with name and parameters so it's easy to understand the action. Return message is shown by return of control. In this sequence diagram driver is look at traffic light then watch for traffic and pedestrians and then do his or her action.

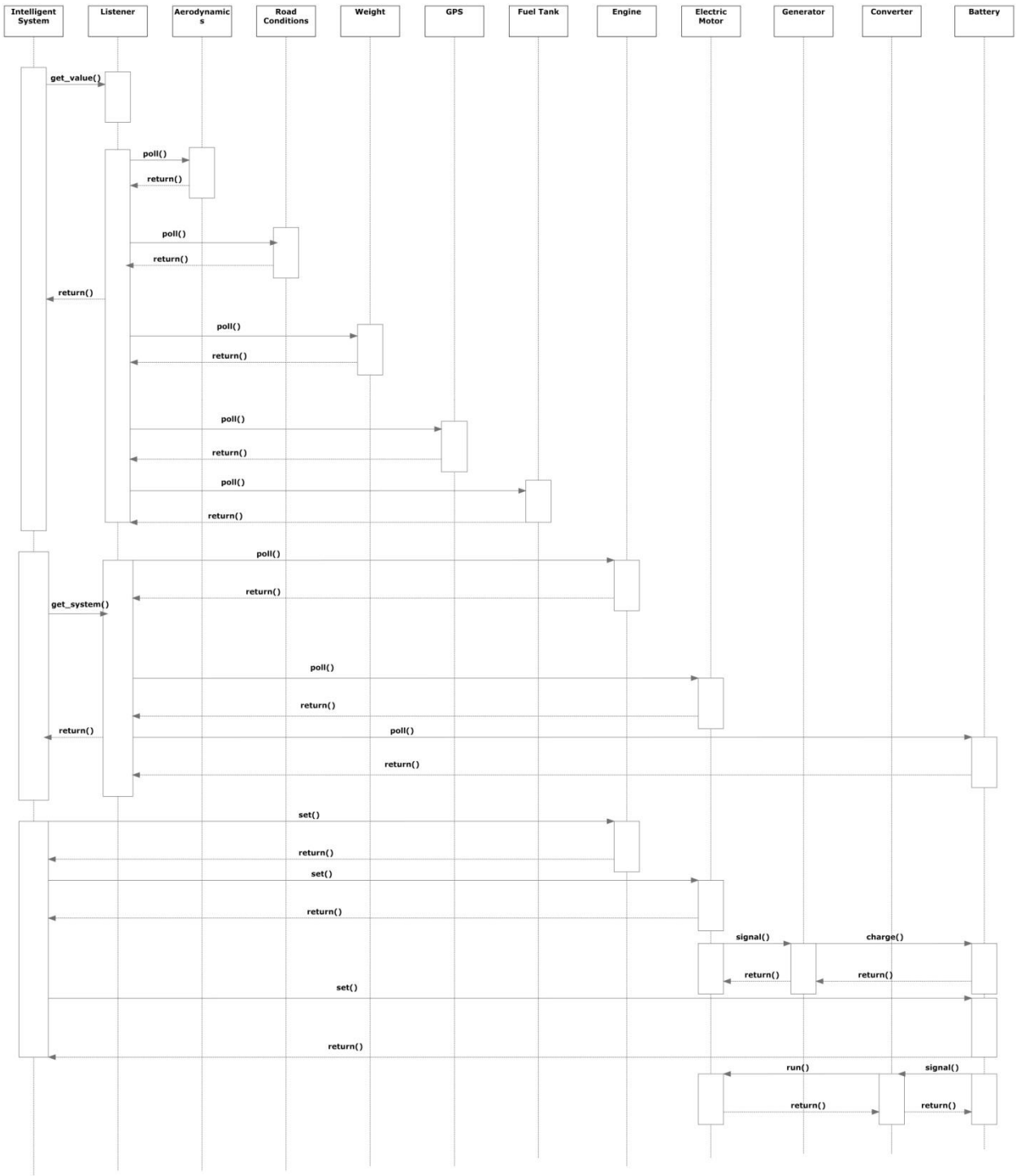


Figure 4-1 Sequence Diagram

5 State-Chart Diagram

State chart diagrams represent the behavior of entities capable of dynamic behavior by specifying its response to the receipt of event instances. State diagrams require that the system described is composed of a finite number of states. Typically, used for describing the behavior of classes, but state charts may also describe the behavior of other model entities such as use-cases, actors, subsystems, operations, or methods. The abstract behavior is analyzed and represented in series of events that could occur in one or more possible states. Each diagram usually represents objects of a single class and tracks the different states of its objects through the system.

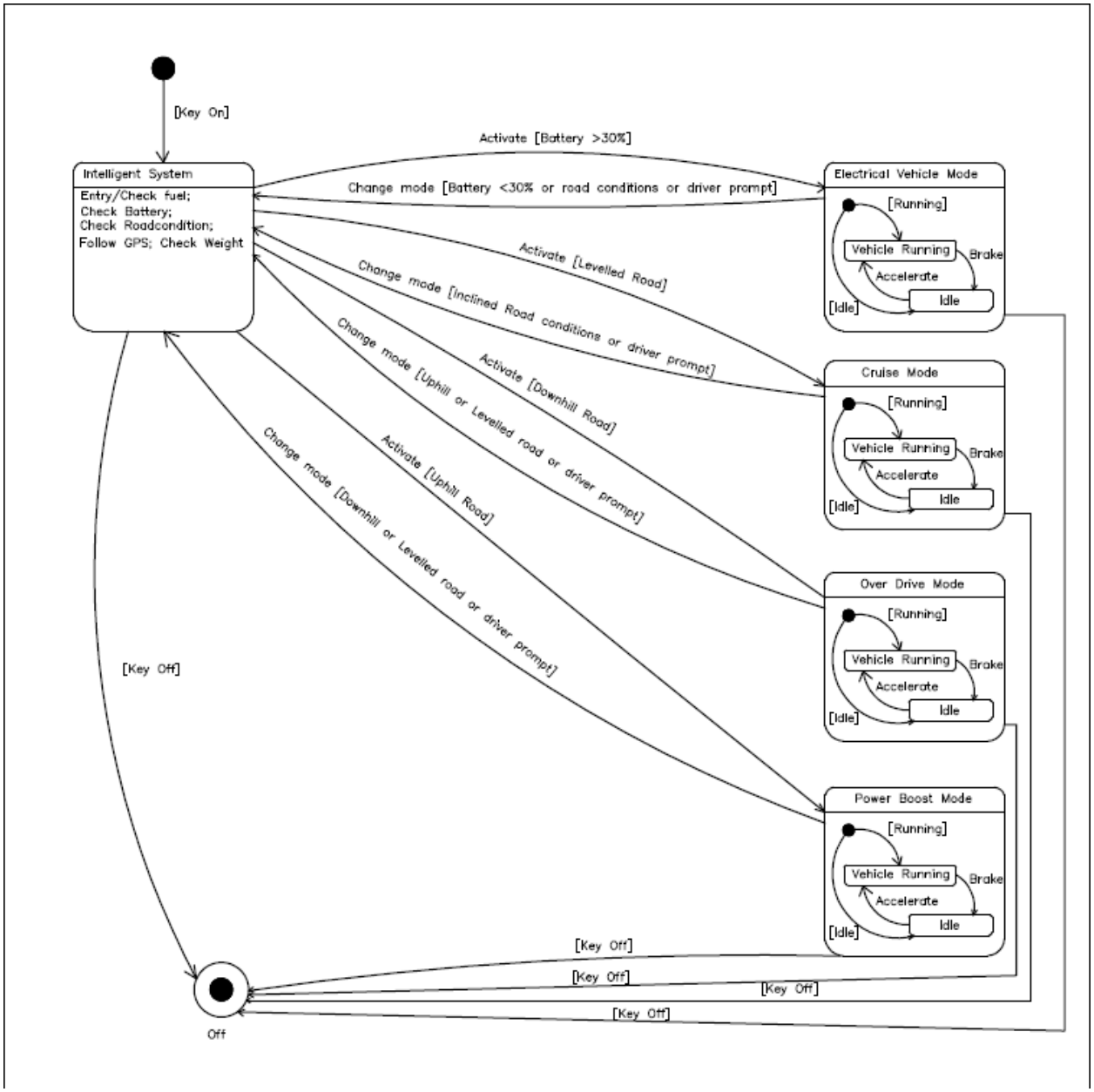


Figure 5-1 State-Chart Diagram

The above state-chart diagram has five states.

1. Intelligent System

2. Electric Vehicle Mode

3. Cruise Mode

4. over Drive Mode

5. Power Boost Mode

6 Class Diagram:

Class diagrams describe a static design of the objects, and their relationships to each other, in an application. During analysis, class diagrams may just be the names of objects and how they interact, but as the design develops the details of each class, including attributes and methods, are displayed in the diagram.

A class is represented by a rectangle divided into three rows. The top row contains the name of the class. The middle row lists the attribute names, while the third row lists the method names for the class. There are two major relationships that can exist between classes: inheritance and association. Inheritance, also known as generalization, describes a super class/subclass relationship. An empty arrow that points from the subclass to the super class represents a generalization relationship.

A line joining two classes represents associations. This association represents a relationship between instances of each class. In an association, one class uses a member of a second class. There are specialized associations, which are aggregation and composition. Aggregation represents a stronger relationship between two objects, and it describes a "has-a" relationship. A white diamond is at the end of the line that points to the class that holds the other class. Composition is a relationship that exists between classes when an instance of one class cannot exist without an instance of a container object. With composition, a black diamond points towards the container class and an arrow points to the contained class.

Class diagrams also contain specific notations about how to describe attributes and methods in the class objects. They also contain information about multiplicity, which indicates how many of each object may be involved in a relationship. Other objects, such as abstract classes, packages, interfaces, and stereotypes, may be represented in a class diagram as well. Overall, class diagrams are used to create a standardized design of the system from the requirements. The

purpose of a class diagram is to depict the classes within a model. In an object oriented application, classes have attributes which are referred as member variables, operations which are referred as member functions and relationships with other classes. The fundamental element of the class diagram is an icon the represents a class.

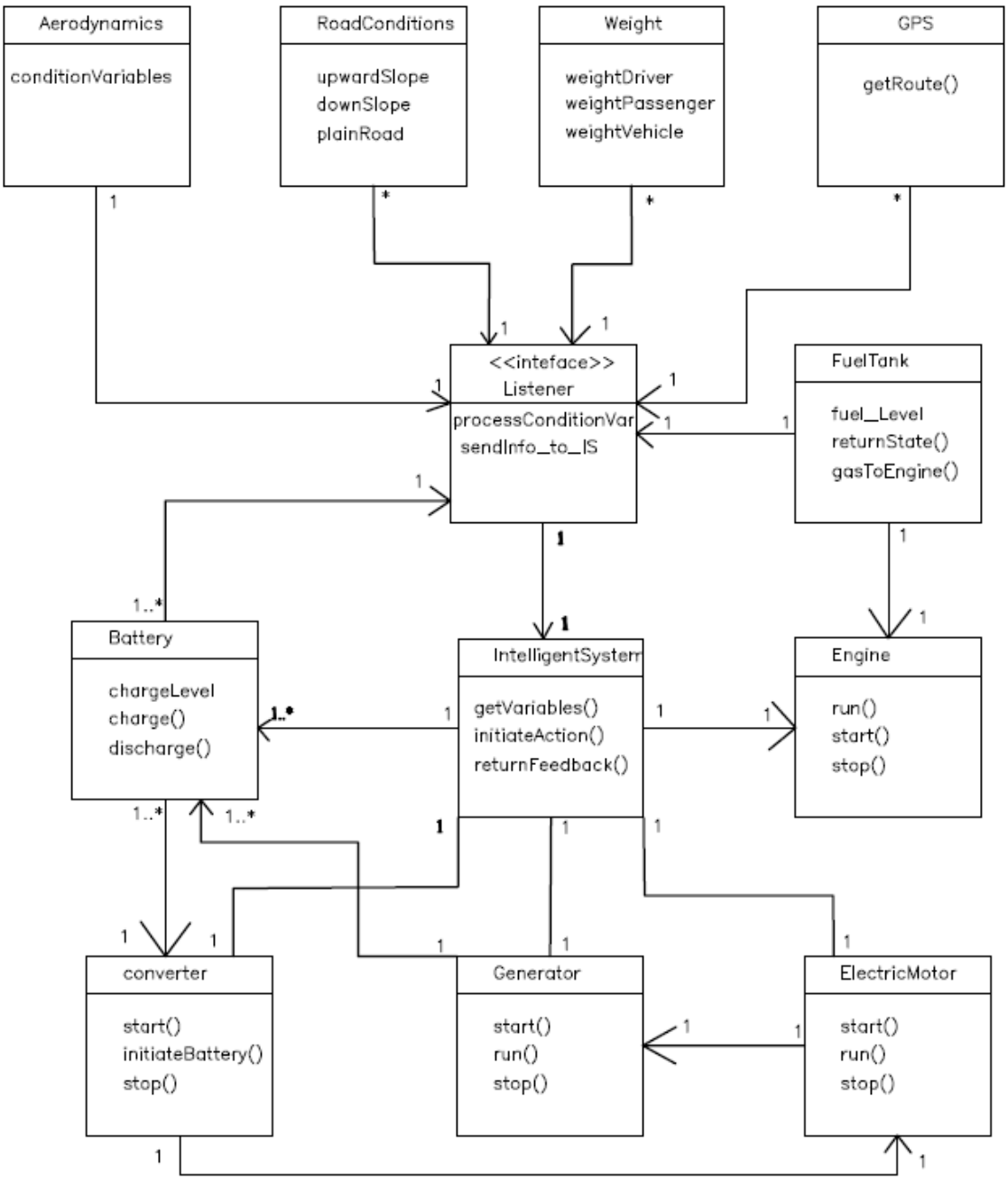


Figure 6-1 Class Diagram

7 Conclusion

We successfully design a hybrid system that supports the process of simulating a hybrid engine system adopting a use-case driven object-oriented approach. Also we successfully designed and simulate a hybrid engine which operates in six different modes. During implementation we assume the average speed, the weight, ideal road condition, ideal engine condition and other ambient factors. In this way, we can use a systematic approach of our understanding of software engineering to any project like this and can be done in easier manner.

8 Sample Output and Screen Shot

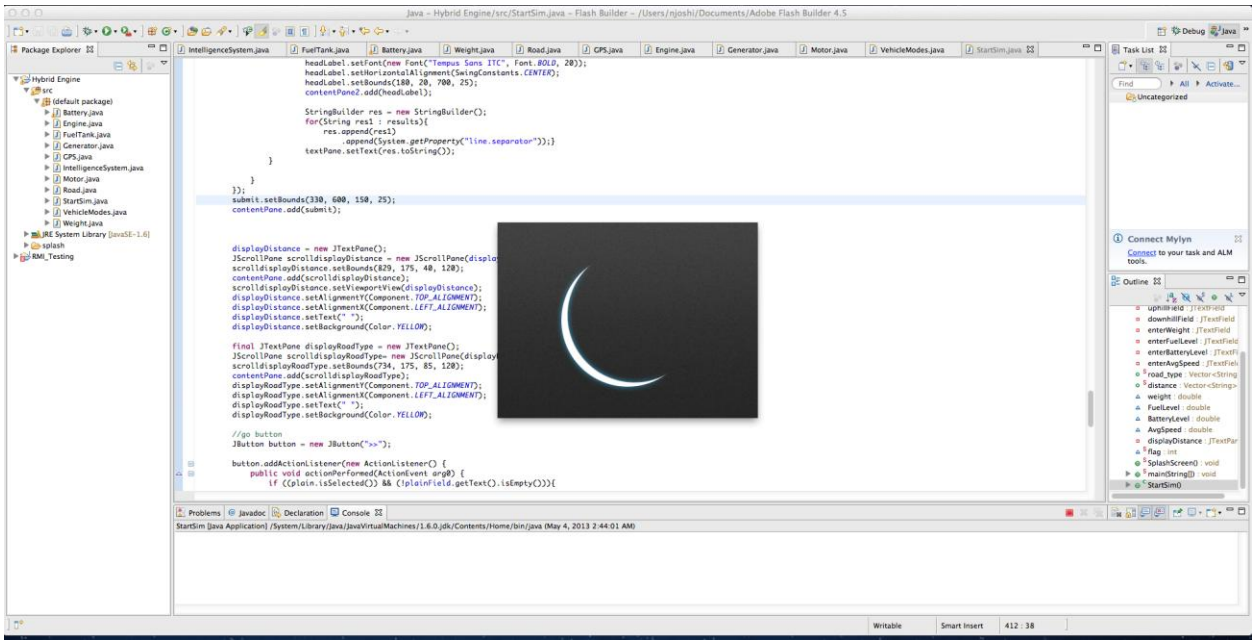


Figure 8-1 after compiling

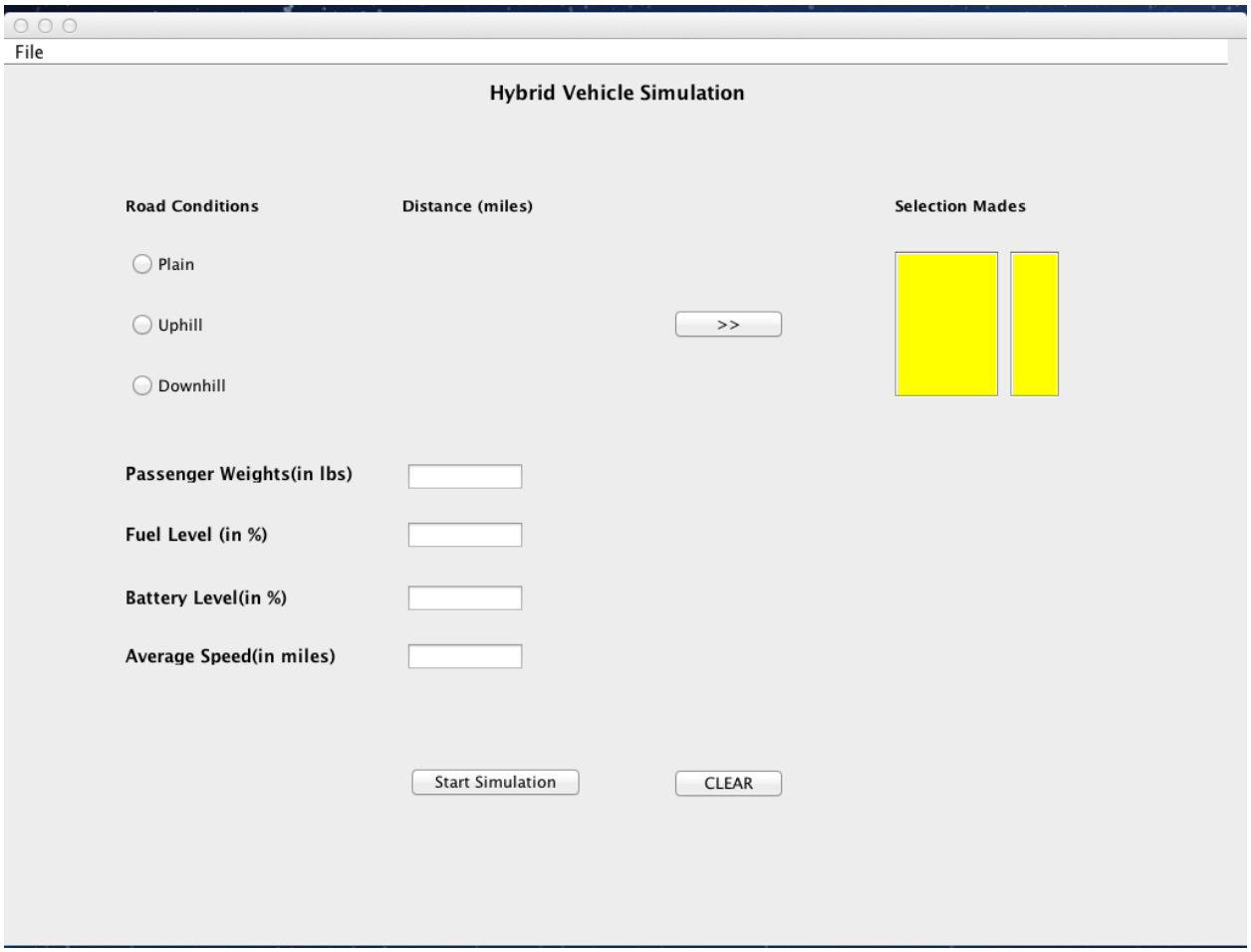


Figure 8-2 Output Screen for simulation

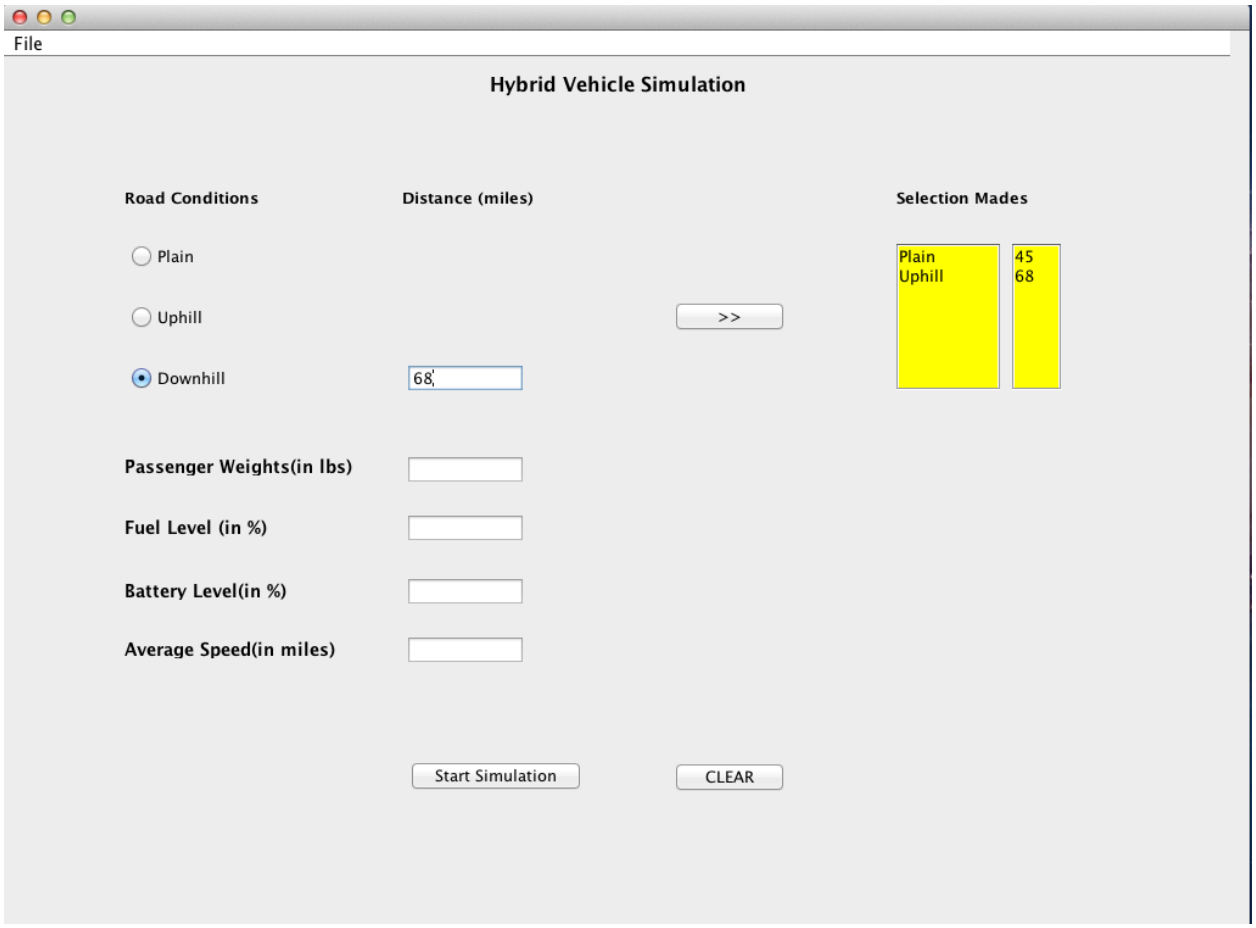


Figure 8-3 Output Screen for entering road condition

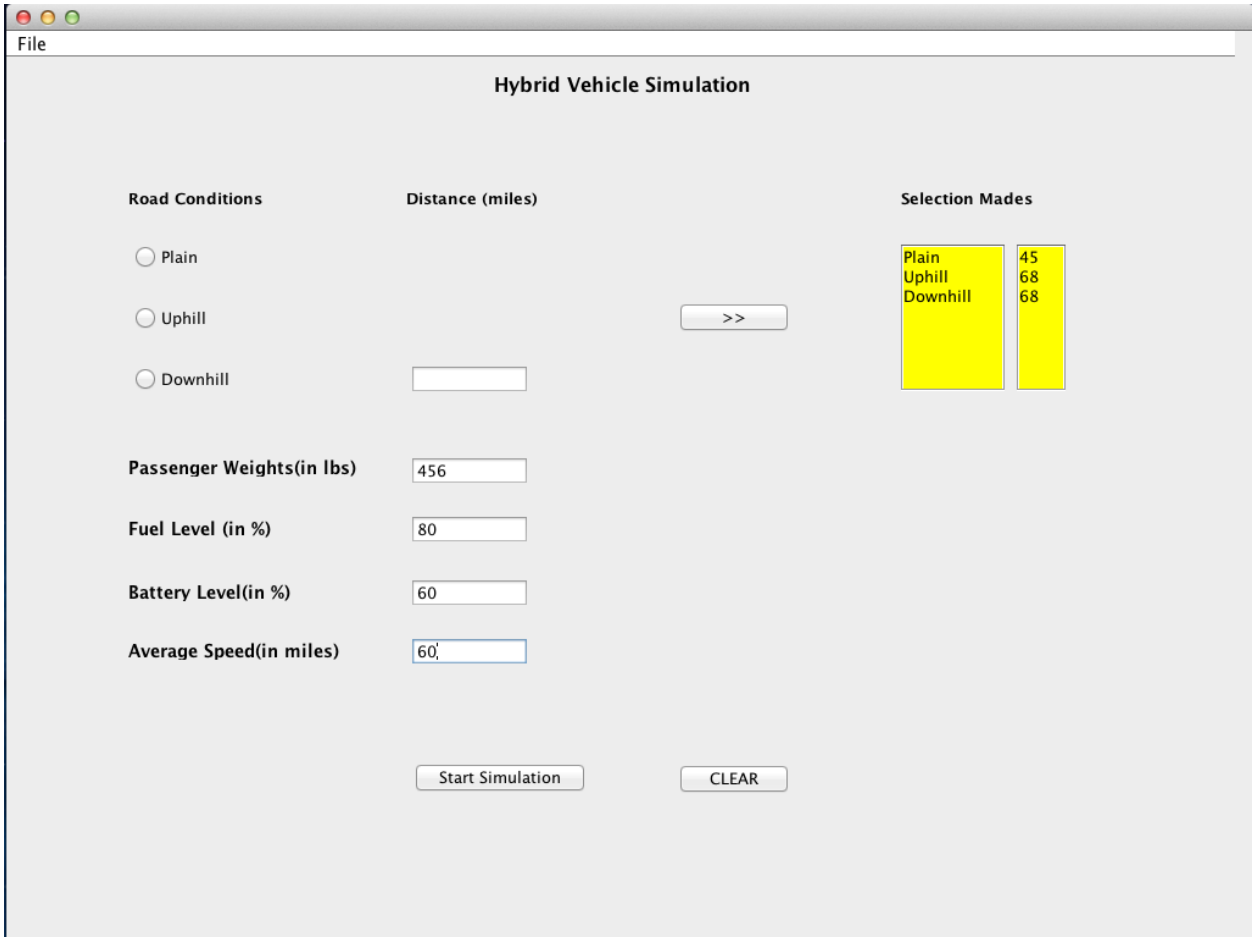


Figure8-4 Output screen for entering vehicle option

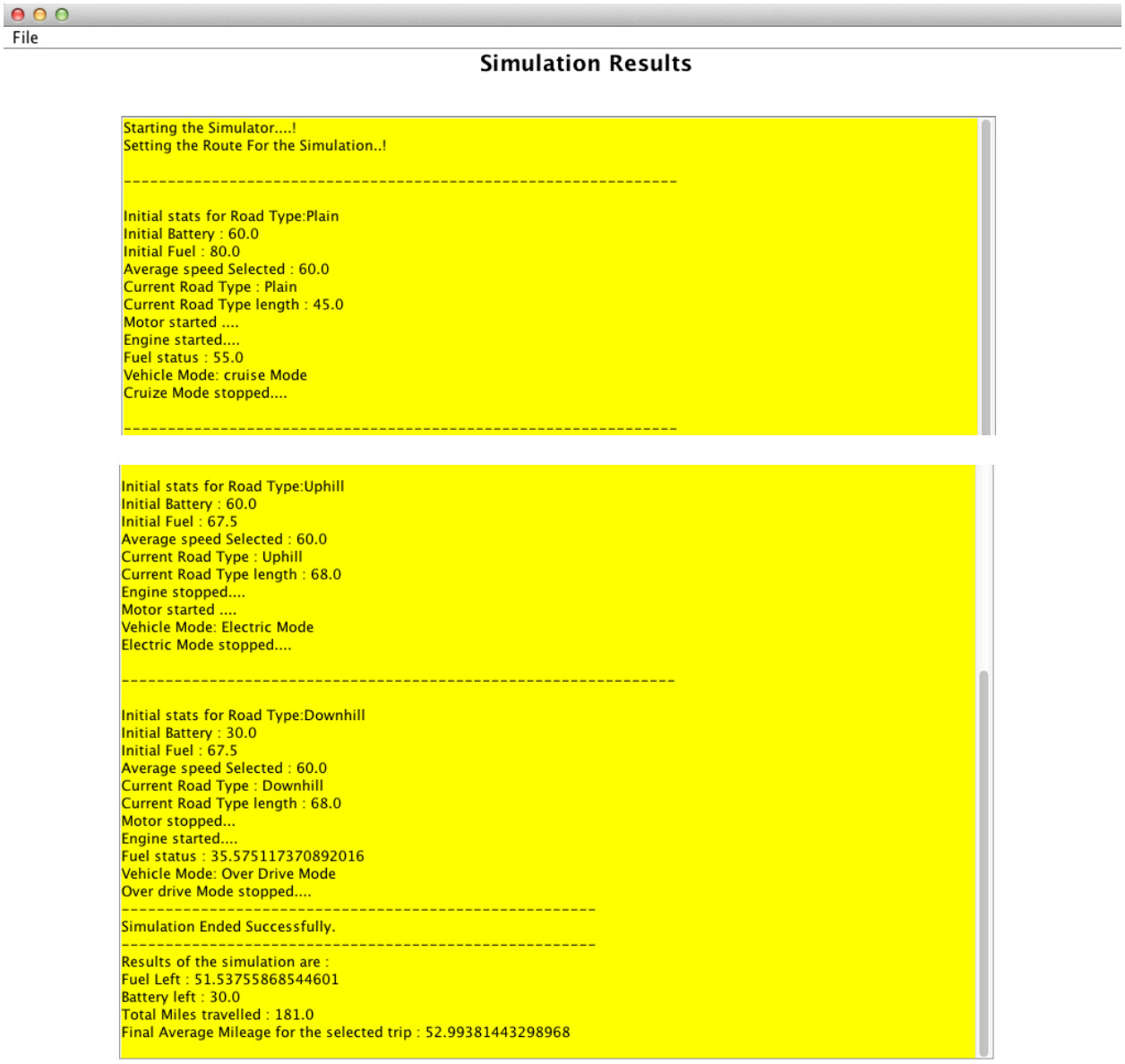


Figure 8-5 Final output

Bibliography

- http://www1.eere.energy.gov/vehiclesandfuels/

- http://web.mit.edu/2.972/www/reports/hybrid_vehicle

- http://www.toyota.com/hybrid/

- http://www.ieahev.org/

-Other reference from Dr. Andrei class notes.