

**Table of Contents**

1. [Introduction 3](#_Toc533070238)
2. [Brief introduction of UML 3](#_Toc533070239)
3. [An Overview of the rural e-bus system 6](#_Toc533070240)
4. [Modeling the static aspects of the system 6](#_Toc533070241)

[4.1 A snapshot of the rural e-bus system 6](#_Toc533070242)

[4.2 Use Case diagram 7](#_Toc533070243)

[4.3 Class Diagram 8](#_Toc533070244)

[4.4 The Static Structure 9](#_Toc533070245)

1. [Modeling the dynamic aspects of the system 10](#_Toc533070246)

[5.1 Sequence Diagram 10](#_Toc533070247)

[5.2 State Chart Diagram 11](#_Toc533070248)

[5.3 Practical Methods 13](#_Toc533070249)

[6. Conclusion 14](#_Toc533070250)

[7. Reference 15](#_Toc533070251)

1. **Introduction**

In this current age of time we see buses all around us but in my perspective of view those buses are not satisfactory in anyway. They just take you from one place to another. In some countries you don’t even know when the bus will come if it will ever come at all. There are number of reasons for this situation. One of the main reasons is the lack of management of the demand and provision of adequate supply of public bus transport that satisfied both the user and operator simultaneously. Inability to understand the qualitative parameters of the transport supply demanded leads to poor response to this situation by both passengers and their communities as well as policy agencies but with the e-bus system we solve these problems.

1. **Brief introduction of UML**

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

Unified Modeling Language (UML) is a general-purpose modelling language. The main aim of UML is defining a standard way to visualize the way a system has been designed. UML are used when designing the system.

UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system. UML helps software engineers, businessmen and system architects with modelling, design and analysis. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. It’s been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

**Object Oriented Concepts Used in UML**

*Class*: A class defines the blue print i.e. structure and functions of an object.

*Objects*: Objects help us to decompose large systems and help us to modularize our system. Modularity helps to divide our system into understandable components so that we can build our system piece by piece. An object is the fundamental unit (building block) of a system which is used to depict an entity.

*Inheritance*: Inheritance is a mechanism by which child classes inherit the properties of their parent classes.

*Abstraction*: Mechanism by which implementation details are hidden from user.

*Encapsulation*: Binding data together and protecting it from the outer world is referred to as encapsulation.

*Polymorphism*: Mechanism by which functions or entities are able to exist in different forms.

**Structural UML Diagrams**

*Class* *Diagram*: The most widely use UML diagram is the class diagram. It is the building block of all object-oriented software systems. We use class diagrams to depict the static structure of a system by showing system’s classes, their methods and attributes. Class diagrams also help us identify relationship between different classes or objects.

*Composite* *Structure* *Diagram*: We use composite structure diagrams to represent the internal structure of a class and its interaction points with other parts of the system. A composite structure diagram represents relationship between parts and their configuration which determine how the classifier (class, a component, or a deployment node) behaves. They represent internal structure of a structured classifier making the use of parts, ports, and connectors. We can also model collaborations using composite structure diagrams. They are similar to class diagrams except they represent individual parts in detail as compared to the entire class.

*Object* *Diagram*: An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behavior when objects have been instantiated, we are able to study the behavior of the system at a particular instant. An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time.

*Component* *Diagram*: Component diagrams are used to represent the how the physical components in a system have been organized. We use them for modelling implementation details. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other.

*Deployment* *Diagram*: Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. We illustrate system architecture as distribution of software artifacts over distributed targets. An artifact is the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations.

*Package* *Diagram*: We use Package Diagrams to depict how packages and their elements have been organized. A package diagram simply shows us the dependencies between different packages and internal composition of packages. Packages help us to organize UML diagrams into meaningful groups and make the diagram easy to understand. They are primarily used to organize class and use case diagrams.

1. An Overview of the rural e-bus system

One of the biggest challenges in this system is some difficult geographic areas because they are unreliable. Rural e-bus system solves all of the problems and also manages the estimated arrival of time of the busses with ease. The rural region in Taiwan with its towering mountains has become the sightseeing hot spots. The 3G/LTE networks are not stable in the mountain geography. The transportation on rural roads will require wireless communication support.

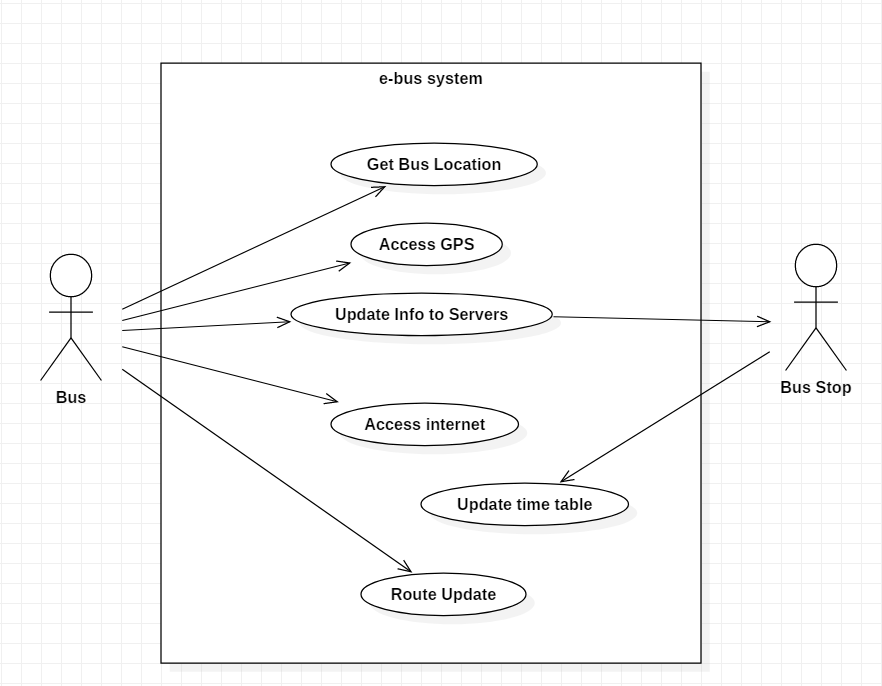
Passengers can make bus reservations with an online vehicle reservation platform system. Through an independent hand power generation mode, the smart traffic sign system can be electrically started up at bus stop. The newest IoT communication technology – LoRa will be utilized for the information portion in order to avoid 3G/4G information dead zones. This technology is particularly suitable for mountainous rural areas.

1. Modeling the static aspects of the system

4.1 A snapshot of the rural e-bus system

The rural e-bus system has the basic function that all bus systems have such as bus stop and bus attributes such as id license and route. But with this rural e-bus system we are doing more than that such as accessing the gps location of the bus and that way we know where is the bus at the specific time. We access the route of the bus at a specific time and that way the passengers know current estimated time of arrival of the bus.

4.2 Use Case diagram



Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system. A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

In this use case diagram we can see some use cases such as get bus location which is used for getting the busses location but while getting the gps location we also need to access gps whether it is by using hardware or accessing the internet. The more connections we have the better. Because if we have more connections we can connect to more satellites. After everything is done finally bus stop gets the location of the bus and estimated time of arrival information and updates the table.

4.3 Class Diagram



In this specific class diagram we can see different classes that connect to each other so we can understand what is going on with the system software-wise. In this diagram we have multiple classes such as hand-cranking power generator, bus, server on bus, route, gps, bus stop, local server, server on bus and remote server. Bus class tells us the id and license of the bus and we also have similar attributes for bus stop but what is different than bus class is that bus stop class also has route attribute to get the route of the bus.

Servers are also a very important part of this diagram because we need to connect to the internet. We connect to the servers by server on bus with LoRa and 4g internet. But if we cannot connect to 4g internet in that specific moment of time we can use other internet connection types such as 3g, 3g+ and even 2g but of course the faster the internet is the better connection we would have to update information about the busses location.

4.4 The Static Structure

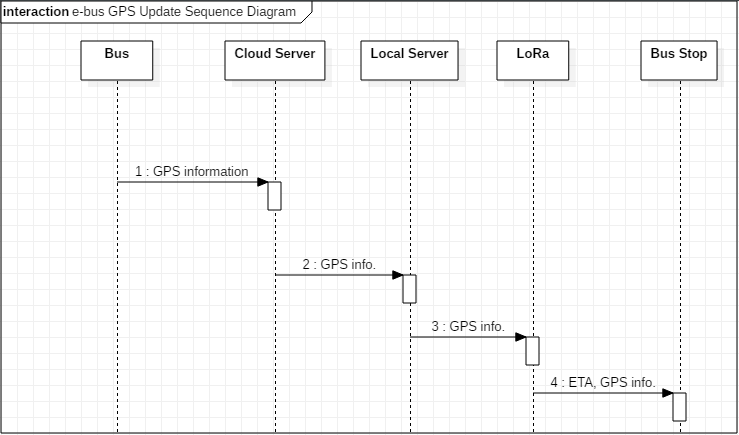
In section 4.3, The class diagram shows an aspect of the system, and gives an overall understanding of the system design when combined together. From the view of object construction, the class diagram describes a solution to the problem as the object architecture. The software design is captured by describing a collection of objects communicating and collaborating to implement a specific function. Objects communicate by sending messages to each other. Objects that share the same responsibilities are generalized into a class.

The class diagram generated from this view grasps the main functional area of the system design, and gives a skeletal describing for the system. From the view of software architecture, much more design and implementation details are captured. Based on the class diagram derived from this view, most of the future design of the software can be figured out. The system architecture view provides the most complicated yet the most delicate description of not only the software, but also the structure of the whole system. Compared to normal software systems, it is very important to know in distributed embedded systems how the system components are working together. After all, every class diagram is just a graphical presentation of the static design view of a system. No single class diagram can capture everything about a system’s design view. Collectively, all the class diagrams of a system represent the system’s complete static design view; individually, each represents just one aspect.

1. **Modeling the dynamic aspects of the system**

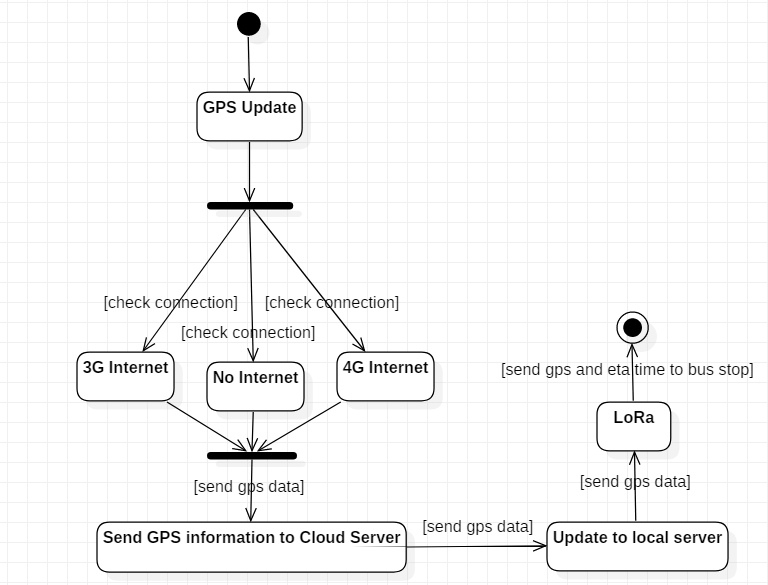
Modeling the dynamic aspects of the system I used the UML which provided me with Sequence Diagrams and State Chart Diagrams to show how this system works step by step and with different aspects of view.

* 1. **Sequence Diagram**



In this sequence diagram we can see how the e-bus system communicates with different servers and that way updates the location of itself through those servers to the bus stop. This is a very important task to do because if not, we can not calculate the estimated time of arrival of the bus to bus stops. Therefore, we need multiple servers such as cloud server and local server that can store and transfer the related gps information to each other so we can surely say where is the bus. Having multiple servers also increases the security of the information so in case of any attack to server or servers we would have a backup storage in some other server to save the data and restore it to the other servers that are under attack at that specific moment of time.

* 1. **State Chart Diagram**



State chart diagrams are used to describe the states of different objects in its life cycle. Emphasis is placed on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately. State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

We can define the practical applications of a state chart diagram. State chart diagrams are used to model the dynamic aspect of a system however, it has some distinguishing characteristics for modeling the dynamic nature. State chart diagrams are used to model the states and also the events operating on the system. When implementing a system, it is very important to clarify different states of an object during its life time and state chart diagrams are used for this purpose. When these states and events are identified, they are used to model it and these models are used during the implementation of the system. If we look into the practical implementation of state chart diagram, then it is mainly used to analyze the object states influenced by events. This analysis is helpful to understand the system behavior during its execution. In this specific state chart diagram we can see that initial state is trying to update the gps location by hardware but we all know that is not enough especially in rural areas because the connection is not strong enough. So after trying to update the gps location by hardware system checks the internet whether the internet is 3g or 4g or even there is no internet at all but it checks for it anyways. Because if there is internet we can connect to more satellites that can help us find where we are. After doing that we send all that gps information to different servers and after sending gps locations to cloud and local servers we connect to LoRa and we calculate the estimated time of arrival of that specific bus to a specific bus stop.

* 1. **Practical Methods**

Making this methods of modeling diagrams takes some afford but it is not impossible. What I used to make diagrams is the UML software. UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development.

Two ways to get started

* Use the online edition of SmartDraw on any computer or tablet
* Start Now

What is meant by UML?

UML stands for Unified Modeling Language. UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices. Improved integration between structural models like class diagrams and behavior models like activity diagrams. Added the ability to define a hierarchy and decompose a software system into components and sub-components. The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite structure diagram, interaction overview diagram, and timing diagram. It also renamed state chart diagrams to state machine diagrams, also known as state diagrams.

1. **Conclusion**

In conclusion, I believe we did a good job by explaining this project to everyone that would like to know more about e-bus system. In the future I hope this system will spread to all rural areas so the people would commute without delays and worries about where is the bus or when will it come. I hope we can solve the issues that stopped this project to spread until this moment which is the lack of management of the demand and provision of adequate supply of public bus transport that satisfied both the user and operator simultaneously. Inability to understand the qualitative parameters of the transport supply demanded leads to poor response to this situation by both passengers and their communities as well as policy agencies but I believe that we will solve these issues when people know more about this project.

In this class I learned many things. A few of the things that I learned is making project reports with proper usage of Microsoft Word by using different methods so explain myself. I also learned how to use table of contents which gave me nightmares until now. I learned how to use the UML software which stands for Unified Modeling Language. In my personal opinion, UML is such a wonderful and useful program that can be learned easily and effortlessly but by learning this useful software we can sketch and organize different kinds of diagrams which can help us accomplish the project or explain our ideas about that project to others. I think e-bus system will be everywhere in the world that will spread and make our lives more convenient in a way that we don’t have to worry about thinking when the bus will come or will we be late to our jobs or school.

1. **Reference**

<https://www.abbreviations.com/UML>

<https://en.wikipedia.org/wiki/Unified_Modeling_Language>

<https://www.smartdraw.com/uml-diagram/>

<https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/index.html>

<http://www.taipeibus.gov.taipei/>

<https://www.cs.cmu.edu/~luluo/Courses/18540PhDreport.pdf> <https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/1252/Complete-Business-Systems-Analysis-Model-UML-Example.aspx>

<https://udn.com/upf/udn/roadjustice_201805/>

<https://creately.com/blog/diagrams/sequence-diagram-tutorial/>

<https://creately.com/diagram-community/examples/t/sequence-diagram>

<https://sourceforge.net/projects/staruml/>

[https://www.tutorialspoint.com/uml\_online\_training/index.asp](https://www.tutorialspoint.com/uml_online_training/index.asp%20)

<https://www.youtube.com/watch?v=UI6lqHOVHic>