TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

RAJESH RAVEENDRAN

SPECIFICATION OF DATA ACCESS OBJECTS AND PERSISTENCE LAYER FOR PROBLEM-SPECIFIC MOBILITY INFORMATION SYSTEMS

Master of Science thesis

Examiners: Prof. Dr.José Martinez Lastra, Dr. Andrei Lobov
Examiners and topic approved by the Dean of the Faculty Council of Engineering Science on 8th August 2016.

ABSTRACT

RAJESH RAVEENDRAN: Specification of Data Access Objects and persistence layer for problem-specific mobility information system
Tampere University of technology
Master of Science Thesis, 68 pages, 6 Appendix pages
August 2016
Master's Degree Programme in Machine Automation
Major: Factory Automation
Examiners: Professor Dr. José Martinez Lastra, Dr. Andrei Lobov

Keywords: Energy Efficiency, Intelligent Transportation System, Persistence Layer, Unified Modeling Language.

Transportation systems are indispensable part of our day to day life. As many people are dependent on transportation systems there are several challenges to be solved. One such challenge is the growing need for making public and private mobility system environment friendly. This increases the need to structure an energy efficient mobility architecture. The main goal of this thesis work is to design and develop a mobility architecture by integrating with prevailing Intelligent Transportation System (ITS) which provides journey options with minimum energy consumption for different types of users.

In order to achieve the goal, a *multi-modal* intelligent transportation architecture is proposed in which the different source of informations are used to improve methods of generating energy efficient journey options. The modeled system considers dynamic traffic information from local traffic management system for calculating the mobility and routing options with *minimum energy consumption*. Some of the other information's are traffic light details, road weather forecast, location of public bus at real time and parking information. In addition to this, to motivate the users the proposed architecture also contains an *incentive-based visualization service* which provides incentives for the users who opts for mobility and routing options which consumes less energy or lowers the cost of CO_2.

In this work, the architecture is modeled using object oriented process and Unified Modeling Language (UML) approach. The services are described using use case and sequence diagrams. Data access object and persistence layer is designed for the proposed architecture using Java language and hibernate framework and finally, the persistence is tested using Junit framework.

PREFACE

The entire research work of this Master’s thesis is performed at *Factory Automation and System Technologies Laboratory (FAST-Lab)*, a research group in the Department of Automation Science and Engineering, Tampere University of Technology, Finland. This thesis work is funded from *MoveUs* EU Project.

I would like to start this section by paying my sincere gratitude to *Prof. Dr. Josè Martinez Lastra* for providing me an opportunity to work at FAST-Lab. I will never forget my diligent supervisor *Dr. Andrei Lobov* for mentoring my work, midnight email replies, working while on vacation, and his constant effort in understanding and motivating me with lots of patience. Now, I thank him from bottom of my heart. I am also very thankful for Anna Florea for helping me. During this thesis work, I faced lots of adversities and complications both personally and professionally (still as a student ☺). In the midst of all chaos, my loving family, true friends (you know who you are), and my dear supervisor pulled me out of the odds and pushed me forward to finish this work. Overall, I learnt how to fight odds of life which are inevitable and keep following the dream.

Finally, this thesis work is not possible without my mother, the women who always believed in my dreams and poured unconditional love. I also realized that my father is the true Dark Knight, thanks to him. I am also thankful to my colleagues Sohail, Xu, Juha, Ahmed, Mehmud, Peyman, Luis, and Borja, for being so helpful and generating an enjoyable atmosphere at FAST.

Tampere, August 03, 2016

Rajesh Raveendran.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
DAO	Data Access Object
EE	Energy Efficiency
EJB	Enterprise JavaBeans
EU	European Union
FCU	Fare Collection Unit
IDE	Integrated Development Environment
ICT	Information and Communication Technology
ITS	Intelligent Transpiration System
JDO	Java Data Objects
JPA	Java Persistence API
JPQL	Java Persistence Query Language
JRE	Java Runtime Environment
JVM	Java Virtual Machine
KAREN	Keystone Architecture Required for European Networks
KPI	Key Performance Indicator
MJP	Multi-modal Journey Planner
OBU	On-Board Units
POJO	Plain Old Java Object
TMS	Traffic Management System
UML	Unified Modeling Language
W3C	World Wide Web consortium
WWW	World Wide Web
XML	eXtensible Makeup Language

1. INTRODUCTION

In this rapidly growing world, energy usage has already raised many concerns about energy supply, energy consumption trend, exhaustion of energy resources and heavy environmental impacts. World market energy consumption was about 524 quadrillion British thermal units (Btu) in 2011 [1]. The commercial sector, industrial sector, residential sector, and transportation sector are the four major energy end-use sectors. According to statistical data provided by Energy Information Administration (EIA) in 2012, the transportation sector alone accounts to 26.6% of global energy consumption [1].

In European Union, over 60 % of the population lives in urban areas, where 40% of CO_2 emissions is caused by urban traffic and 70% of emissions is caused by other pollutants arising from road transport [2]. And nearly 12% percent of the total CO_2 emissions is produced by vehicles, consequently in the new vehicle regulations CO_2 emissions should be reduced 40% in new vehicles from 158.7g/km in 2007 to 95g/km of CO_2 in 2050 [3]. In Finland specifically, the CO_2 emissions due to transportation sector is second highest which accounted for 22.1%, in 2011 [4]. And since 2000 it is increasing moderately from 21.4% [4]. Hence, it is imperative to adopt a transportation system which use energy efficiently.

Transportation sector undeniably bolster social development and economic growth by increasing mobility options and improving accessibility (or reachability) to people, resources, and markets. Transportation is a necessary infrastructure for our modern society because its performance has a direct impact on individual mobility, social and economic growth of all countries. Also the counties competitiveness, strength, and productivity depends on the transportation infrastructure and its performance [5]. Information and Communication Technology (ICT) has been fundamental to transport. Today's transportation network has lack of interoperability and are not integrated to provide multi-modal services and inter-modal service. Several impacts of traffic and transport, and the associated costs, can be distinguished, such as economic, social and environmental impacts, for both current and future generations. In [6][7][8][9] the authors have mentioned that with an innovative Intelligent Transportation System (ITS) architecture which comprises concept of shared vehicle systems can potentially reduce the CO_2 emission, alleviate traffic congestion, improve air quality, and make surface transportation more efficient.

1.1 Background

With increase in world population, energy efficiency became one of the important issue in transportation sector. The current scenario of transportation sector demands the need for energy efficient Intelligent Transportation Systems (ITS) due to increased population, constant change in population density, traffic congestion, accessibility, urbanization, increased motorization etc.

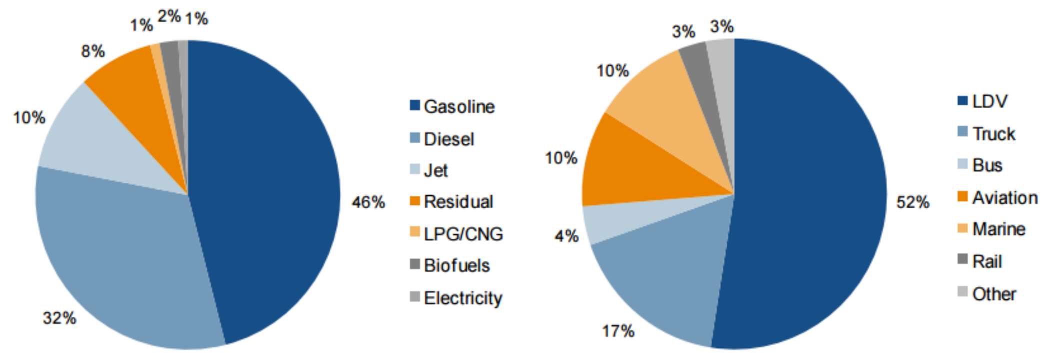


Figure 1. *2010 Transport energy by source and by mode [10].*

Transportation accessibility (or reachability) refers to how easy to get to a destination form one place. The transportation accessibility determines the social and economic intimacy between one city and the destination city under the ITS. In that case the regional transportation accessibility index is relatively low, less people will go out and the places they go are within a smaller range. The potential movement of the people living in the city will come into outing obstacles. The accessibly reflects the chance and potential of economic and technical communication between one region and other relative regions. So the transportation accessibility of one city to its adjacent cities will mirror and affect its economic development.

Presently, most of the traffic problem and accessibility is often based on a lack of information. Of course when making travel choices, road users constantly combine various sources of information and expectations of traffic conditions, but they can hardly obtain data about the global situation. Thus, they have only a partial and inaccurate knowledge about the traffic conditions in the network and it is not possible for them to coordinate their behavior with the other, e.g., change their departure time to relax the situation. Additionally, the most useful information to a driver is predictive information. Since, it allows the road users to estimate the travel time of their trips under consideration of future developments of the traffic conditions. Hence, the need for multi-modal service structure arises. In a multi-modal service system, the traveler can travel from the origin to the destination via multiple transportation modes with unit costs. A *Multi-modal service* proposes journey options with one or more transport services providing solutions from source to destination with different modes of transportation system at a given arrival time and predefined conditions.

1.2 Problem Definition

1.2.1 Justification of work

With excessive use of private mode of transportation in big cities there is an increase in traffic congestion, transportation accessibility, lack of parking space, and more energy usage. Public transportation can be considered as an economic alternative but still it will lack accessibility and flexibility of private transportation system.

Hence there is a need for intelligent transportation architecture with focus on *multi-modal services*. Intelligent transportation technology can play an important role in making these systems user-friendly, easy to manage, and energy efficient. The architecture described in this work contains innovated techniques for calculating multi-modal journey options, Energy Efficiency (EE) / CO_2 assessment of journey options, and incentive-based visualization service. It considers dynamic traffic information from local traffic management system for calculating the mobility and routing options with minimum energy consumption. Some of the information's are traffic light details, road weather forecast, location of public bus at real time, and parking information.

In this work, a travel advisory system is also proposed with the aim to provide guidance for travelling route, suggestions for journey options with other modes of transportation such as public transportation, bike transportation, car sharing or even carpooling [11], and generating incentives for travelers who chose other recommended modes instead of using their own vehicle.

1.2.2 Problem Statement

Problem addressed in this thesis is "*how to create a mobility architecture targeting increase in energy efficiency for different types of users?"*.

1.3 Work Description

1.3.1 Objectives

Objective of this thesis work is to model the use cases and implement data persistence for a multi-modal intelligent transportation architecture so that it can help in providing necessary information to improve methods of generating energy efficient journey options. The proposed architecture also contains an incentive-based visualization service which provides incentives for the users who opts for mobility and routing options which consumes less EE/CO_2. The architecture developed in this thesis work is performed for Tampere region of Finland.

1.3.2 Methodology

In this thesis work, the dynamic traffic data provided by the local traffic center such as traffic situations, weather conditions, and status of traffic signals and other data related to mobility were taken into consideration to create an architecture which provides journey options with minimum energy consumption for different types of users.

The objectives of this thesis is achieved by performing a series of steps in order. Firstly, the researches on various other ITS architectures and survey in the domain of energy efficiency pertaining to transportation sector were made. After this research work, review of software modeling techniques was performed. It also includes the review of system integration methods using selected object oriented process and UML technique for design and development was done. Performed a complete study of prevailing mobility and ITS infrastructure in Tampere region of Finland.

After this, a thorough study of different approach and technologies for designing and implementing the data persistence layer was done. Based on the selected approach and technologies the design was modeled and implemented.

1.4 Thesis Outline

The future chapters are organized as follows: Chapter 2 introduces theoretical background of the thesis. It encompasses extensive review of Intelligent Transportation System, energy efficiency in the context of transportation systems, and also theoretical background of well-established methodologies. Chapter 3 discusses the approach to solve the problem mentioned in the previous chapters. It illustrates the existing mobility and ITS infrastructure; it also explains about the technologies used in this thesis, and the system design for the proposed services. The implementation is extensively elaborated in Chapter 4 along with the results. Finally, chapter 5 contains the conclusion of the research work and future research direction and extension in this domain.

2. THEORITICAL BACKGROUND

2.1 Intelligent Transportation System

Today’s transportation systems are not interoperable and are not integrated to provide multi-modal services. It is also becoming more congested and there is a growing need to adopt to improvised transportation system architecture without exploiting the existing assets. The advancement in information technology leads to new doors such as Intelligent Transportation System (ITS). This system helps in monitoring the current scenario, predicting the future, additional traffic information, and provides energy efficient mobility options proactively on a large scale.

The following sections introduces the concept of ITS, describes other well-established ITS architectures and then discusses about different *Multi-modal Journey Planner* (MJP) model from user point of view.

2.1.1 ITS Conceptualization and Design

“Intelligent Transportation System” term can be defined as to integrated telematics, communication, control, and automation technologies that significantly contribute to improve the quality of transportation services. In other words, it is an advanced technology which aims to provide innovative services pertaining to diverse modes of transport and traffic management. It enables different kinds of users to access plethora of information to make journey safer and more coordinated, and use a multi-modal and intermodal transport networks.

It is the integration of communication and control systems that helps authorities, operators, and individual people to be informed better and support decision making [12]. What is new regarding ITS is that it encompasses a broad range of wireless and wire line communications based information and information processing, control algorithm, electronics and other technologies. When integrated into the transportation system's infrastructure, and in vehicles themselves, these technologies relieve congestion, improve safety, improve business operations by sharing and monitoring real time information, and to enhance productivity. ITS applications include systems for traveler information, traffic management, freight management, parking assistance, route guidance, electronic payment and driver assistance.

The authors [13] described the functionalities of a powerful multifunctional data-driven

Intelligent Transportation Systems (ITS) with contrast to a conventional technology-driven system. One of the key functional components is *Multi-modal Evaluation Criteria*. The performance of some applications of ITS cannot be evaluated with one criteria. After detailed analysis, it is concluded that with the promotion of multi-modal transportation with efficient collaboration of automobiles, railway, and water transportation for cargos in Japan, a significant reduction of carbon emission can be achieved [14]. Even though in this thesis work we have different modes of transportation the analysis is scalable for proposed multi-modal transportation also. During this thesis work other notable existing architectures has been referred [15][16][17]. Among them [17] FRAME Architecture, an EU funded project Keystone Architecture Required for European Networks (KAREN) shares common interest for developers who are interested in European ITS.

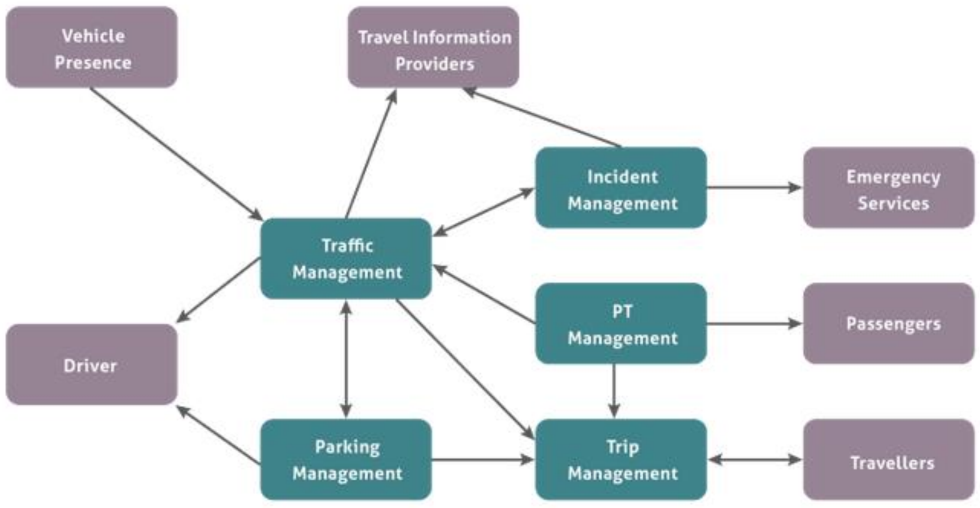


Figure 2. *The components of an ITS – Functional viewpoint [17].*

The Functional view of FRAME is shown in Figure 2 and the entire ITS architecture is designed in complex hierarchical *Data Flow Diagrams.* The ITS is framed with central *Traffic Management System* (TMS) which coordinates the other system and its data. Figure 3 shows the complex highest level of data flow diagram of KAREN.

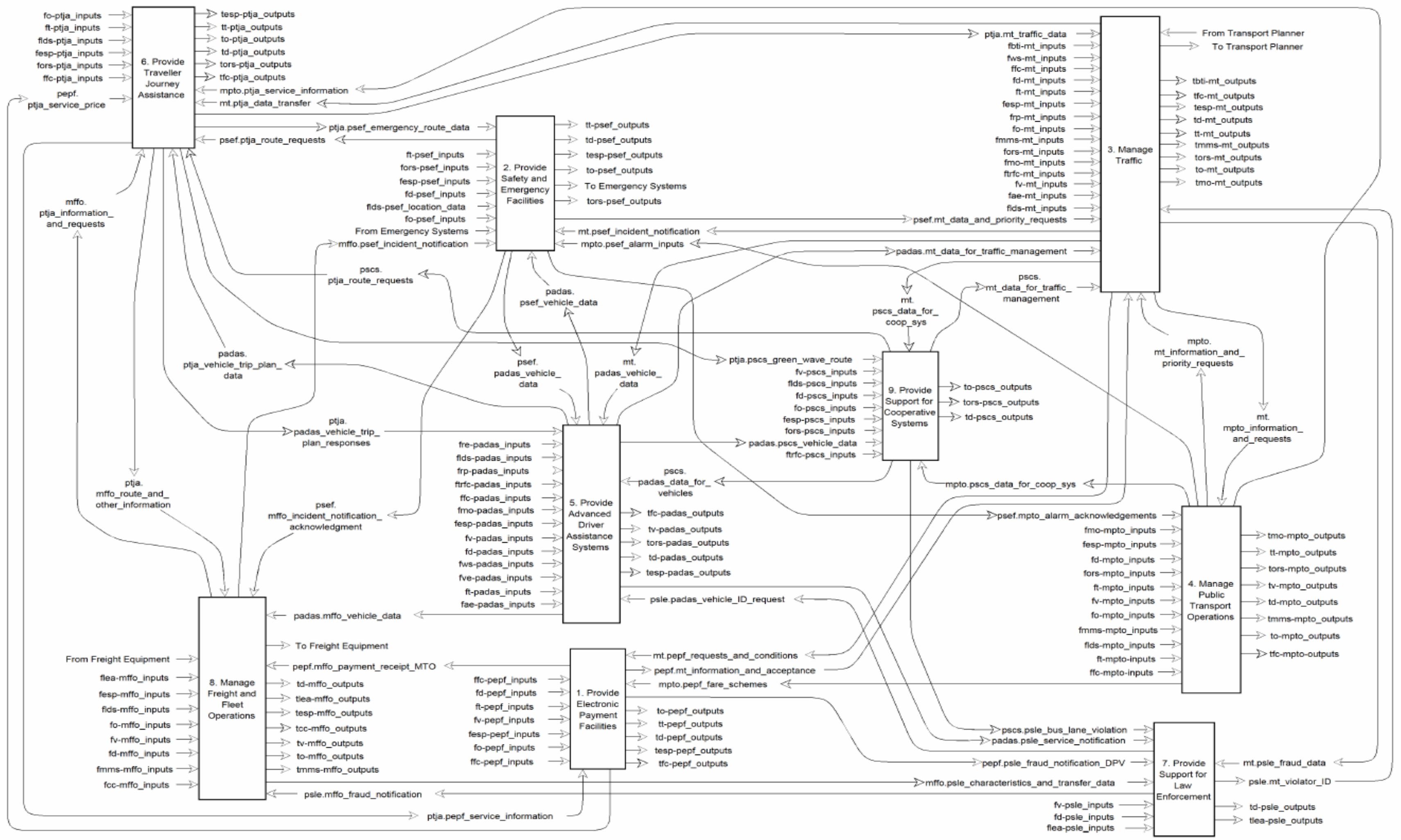


Figure 3. *DFD0, the highest level of Data Flow Diagram [20].*

These traditional ITS architecture which are developed in data flow diagrams have intensive rigidity, non-reusable property, large size, and complexity. An alternative for data flow diagrams is using object oriented process and UML. In [19] the authors have described an approach for design and development of ITS architecture and its system by using object oriented process and UML. In this thesis, the design of architecture is implemented using object oriented process and UML. This approach not only reduces issues with data flow ITS [19] but also provides an overall overview of the system for the developer. According to the authors [21] UML-based systems integration modeling technique is an instructive attempt for the systematic design and development methodology in intelligent transportation management system. Hence, based on this research it is advisable to implement ITS models based on UML.

The Figure 4 illustrates one of the key service, *Multi-modal coordination* in Transit Management of National ITS architecture [15][18]. The service aims to provide Multi-modal coordination between transit agencies (multi-modal transit) and the main advantage of this service is that it increases traveler's convenience at transit transfer points and clusters such as a collection of stops, stations, or terminals where transfers can be made conveniently. It also coordinates with Parking Management System. The service particularly increases the operating efficiency but fails to generate energy efficient transit transfer. This work attempts to generate transits with different modes by calculating energy consumption. Finally, a multi-modal journey options which is embedded into the architecture can improve the effectiveness of existing ITS architecture.

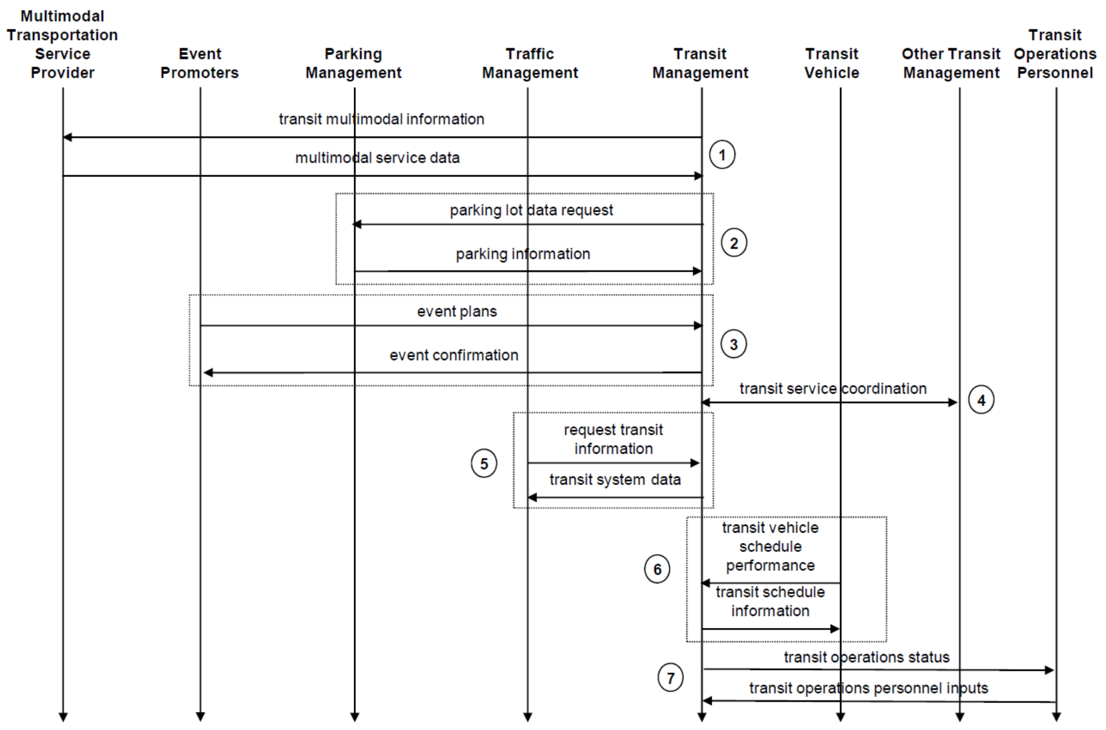

Multimodal Transportation Service Provider
Event Promoters
Parking Management
Traffic Management
Transit Management
Transit Vehicle
Other Transit Management
Transit Operations Personnel
transit multimodal information
multimodal service data
1
parking lot data request
parking information
2
event plans
event confirmation
3
transit service coordination
4
request transit information
transit system data
5
transit vehicle schedule performance
transit schedule information
6
transit operations status
transit operations personnel inputs
7

Figure 4. *Multi-modal Coordination [18].*

In a research [22], an approach is derived to estimate the energy to be used along one specific route and then compute the most energy-efficient route for a given origin and destination using multi-modal transportation architecture. In this approach, real-world restrictions are incorporated and also well know and efficient Dijkstra’s algorithm [23] is used for computing the shortest route. Also this minimum energy routing was extensively tested in real-world condition with positive result [24][25][26]. Hence, it is evident that multi-modal transportation system which is proposed in this thesis work can create energy-efficient mobility options. However, the thesis work contains only design and development at architecture level, the next stage is to test the system with Dijkstra’s algorithm for selected origin and destination given the different mobility options. This stage is beyond the scope of this work.

2.1.2 Multi-modal Journey Planner - User Experience

The authors [27] proposed a model for multi-modal journey planner which connects the networks by using transfer links for which pedestrian users play an important role. The authors [29][30] proposed a Multi-modal Transport Advisory system by integrating data from different sources such as public transportation systems, car, and bike sharing or carpooling based on a domain ontology for Public Transportation. This approach is beyond the scope of this thesis but this approach can be used for future extension.

Genetic algorithm used by authors [28] to construct sequence of transport modes by considering journey plan modes such as walk, bus, underground, and taxi. However, the individual modes of transport such as bike, shared bike, and car are not used. The above Table 1 compares the different *Multi-modal Journey Planner Model* based on different criteria.

Table 1. *Comparison of Multi-modal Journey Planner Models*

Journey Planner Model	Approach	Modes and Information	Benefits	Drawbacks
Multi-modal Transportation Advisory System [29]	Ontology, Dijkstra algorithm, UML	Bus, Tram, metro, Train, and Ferry.	Provides transnational networks of regional and national and provides tourist guide.	• No energy consumption optimization. • No private vehicle included.
Multi-modal Transport network modal [27]	Dijkstra algorithm	Foot, bike bus, and car.	Public and Private network integrated, and includes pedestrians.	• Slow Computation. • No energy consumption optimization.
Multi-modal Route Planner [28]	Genetic Algorithm	Foot, Bus, Underground, and Taxi.	Optimal, feasible, more intelligence, and personalized routes between origin, and destination.	• Complex. • No energy consumption optimization. • Soft transports such as electric vehicle and bicycle, and also private vehicles are not considered.

2.1.3 Finland ITS

The world's first national ITS strategy covering all modes of transport is Finland's Strategy for Intelligent Transport 2009 [31]. It also attracted international attention and received the European Commission's eSafety Forum's Policy Award in Brussels in autumn 2010. In 2013, the strategy was later revised elaborately.

The Transport Policy Report (approved in April 2012), promotes the creation of a transport system centered on the level of service. In this system, a public sector client defines the level of service required, and service providers are given greater freedom to meet these requirements through the technological means of their choosing. In the future, all levels of the transport administration will reflect this change in thinking. This thesis work is an example of one such drift in thinking. The freedom provided by this policy allows the developers to design new transportation architectures focusing future needs. Overall, ITS Finland promotes the globalization efforts of its members - particularly of small and medium-sized companies through ITS Europe Congress and also export network activities. In addition to this, it acts as an advisory community of experts, which represents its members in implementing the strategy and – in cooperation with other stakeholders – monitors international developments and keeps its sector well-informed on these developments.

This thesis work is a by-product of pilot project for *MoveUs*[1], mainly in Tampere region. The main target of Tampere mobility system is to pioneer-ship in climate change. One of the strategic targets of the city represents a 40% reduction in CO_2 by 2025, compared to the 1990 levels[2]. It is highly feasible for the transportation sector to achieve this goal, by shifting the modal split in favor of cycling and public transport. The most important step towards effectively managing journey choices is to quantify the environmental footprint of available choices.

Having said this, the main issue of focus for Tampere is to enable integrated awareness of energy/CO_2 consumption of all possible journey (i.e. mobility & routing), per user. The general objective is to reduce the environmental impact of the urban traffic in Tampere allowing fluent, environmentally friendly and safe flow of public and non-motorized traffic.

2.2 Current Technologies

In order to achieve the objective of the thesis, a review of methodologies and technologies pertaining to persistence interfaces between Java application objects and stored data is made extensively.

There are many methodologies and technologies for implementing the persistence layer and data access objects. In this section, some of the well-known methodologies and the related technologies are discussed. Also a brief introduction to hibernate framework is illustrated with its architecture, which is used later in this thesis work during implementation.

[1] MoveUs, Available Online: http://www.moveus-project.eu/
[2] ITS Factory, Available online: http://wiki.itsfactor<y.fi/index.php/ITS_Factory_Developer_Wiki

2.2.1 Java Data Objects

Java Data Object (JDO) was initially developed as a Java Specification Request 12 (*JSR12*)[3] under Java Community Process and the current *JDO 3.0.1* is *JSR-243*[4]. It is an interface-based API for selection and transformation of transactional persistent storage data into native Java programming language object. This specification provides unified, simple and transparent persistence interfaces between Java application objects and data stores. It is also an interesting alternative to entity beans.

2.2.2 Enterprise Java Bean

Enterprise Java Beans (EJB) is the server-side component architecture, which provides infrastructure for developing highly scalable and robust enterprise level applications [32]. And they are deployed on Java Platform, Enterprise Edition (Java EE) complaint application servers such as JBOSS, Web Logic etc. It enables rapid and simplified development of distributed, transaction, secure and portable applications based on Java technology.

The shift from *EJB 2.0* to *EJB 3.0* is a big change and it made *EJB* based application development a lot easier [33]. The Table 2 elaborated the difference between *EJB 2.0* and *EJB 3.0*.

Table 2. *Comparison of EJB 2.0 and EJB 3.0 technologies.*

EJB 3.0	EJB 2.0
Annotations and defaults reduces a lot of cumbersome code.	Developers have to create lengthy, complicated and deployment descriptors in the application that uses EJB technology.
Persistence entities are simple concrete, and plain old java object (POJO) classes that developers can run as they would any other simple java technology classes.	Developers have to implement the various abstract classes, interfaces, and interface methods whether they use them or not.
The API provides capabilities, such as inheritance and polymorphism.	Those capabilities are not available in the *EJB 2.0*.

[3]Java Specification Request (JSR)-12, Available online: https://www.jcp.org/en/jsr/detail?id=243
[4] Java Specification Request (JSR)-243, Available online: https://www.jcp.org/en/jsr/detail?id=12

2.2.3 Java Persistence API

Java Persistence API (JPA) is a Java application programming interface specification that describes the management of relational data in Java applications. The JPA exactly was described as part of the EJB 3.0, which is a replacement to the EJB 2.1 Entity Beans specification. The Figure 5 describes an overview of JPA in a Java application. This specification offers the developers much easier way to access and manipulate relational data in the Java application. In this thesis work, the persistent element of the system modeled in UML is later converted into following JPA persistence standard. Because UML does not provide resources to model database persistence [34]. It consists of three main areas:

- APIs defined in *javax.persistence* package
- Java persistence Query Language
- Object/relational meta data

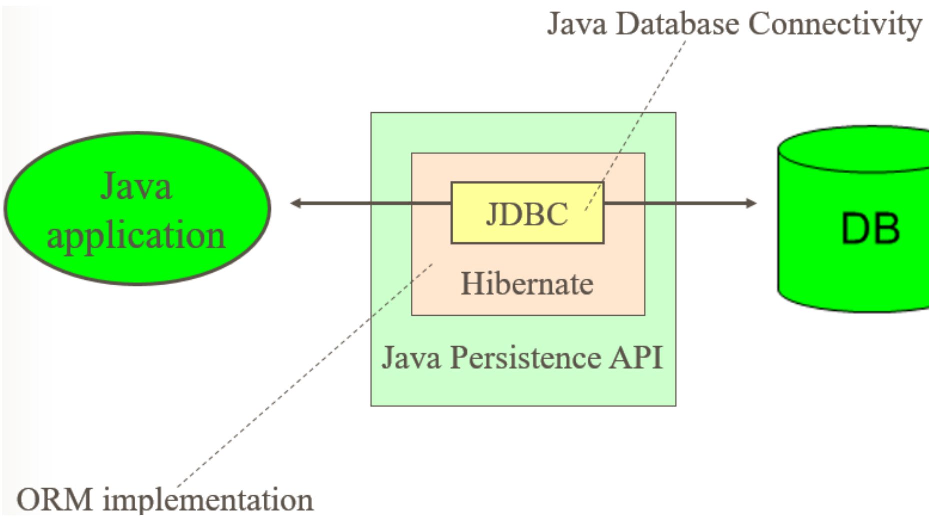


Figure 5. *An Overview of JPA.*

And some of the main concepts of Java Persistence API are described below.

Entity

An entity represents the table in a database which in turn represents a class. And each instance of entity corresponds to a row in the database.

Entity Relationships

The entity relationships are the same as the relationships between tables in a database. The four main relationships are:

1. ***One-To-One:*** Each instance of the entity (row) is related to single instance of another entity (to a single row of another table).

2. ***One-To-Many:*** An entity instance can be related to more than one instance of another entity. But an instance of the other Entity can only relate to one instance of the first.
3. ***Many-To-One:*** Multiple instances (rows) of an Entity can be related to one instance of another Entity. But an instance of the other Entity can relate to only one instance of the first Entity.
4. ***Many-To-Many:*** An instance of one Entity relates to many instances of another Entity and vice versa.

Entity Class

Each Entity Class has instance variables. The class has *accessor* and *mutator* methods that follow the Java Beans convention for naming the methods. The persistent state can be accessed either through the instance variables or through the getter methods. The annotations are defined in *javax.persistence*, so you'll need to import that package.

Entity Manager

They are used to manage the entities. Objects of Entity Manager are represented by an instance of *javax.persistence.EntityManager*. Each object is associated with a persistent context. These objects are used to remove the persistent entity instances, find Entities by using the primary key and allow queries to be run on entities.

Persistence Unit

It defines all Entity classes that are managed by *EntityManager* instances within an application. A name should be provided to describe the Persistence Unit in your application. A Persistence Unit is defined in the *persistence.xml* configuration file. Location of the file within the application in the server varies depending dependent on if the client is a client application or a web application.

Java Persistence Query Language

Java Persistence Query Language (JPQL) is defined in *JPA* specification. It is used to create queries against entities to store in a relational database. *JPQL* is developed based on *SQL* syntax. But it won’t affect the database directly. *JPQL* can retrieve information or data using SELECT clause, can do bulk updates using UPDATE clause and DELETE clause. *EntityManager.createQuery()* API will support for querying language.

The below Table 3 shows the annotations used in JPA and its equivalent UML elements.

Table 3. *Annotations of JPA and Equivalent UML Element.*

JPA	UML Element
@Entity	class
@Inheritance(Strategy)	Generalization
@(One/Many) to (One/Many)	Property
@Embeddable	Class
@Transient	Property
@Embedded	Property
@Id	Property
@IdClass	Class
@EmbeddedId	Property
@Column	Property
@Version	Property
@Enumerated	Class
@Lob	Property
@Temporal	Property
@AttributeOverride	Class, Property
@OrderBy	Property
@DiscriminatorColumn	Class
@SequenceGenerator	Property/Class
@TableGenerator	Property/Class
@AssociationOverride	Generalization

Some advantages of *JPA are*:

- This technology requires small number of classes and interfaces.
- It gets rid of lengthy deployment descriptors by replacing with annotations.
- *JPA* addresses most typical specifications through annotation defaults.
- *JPA* supports easier, cleaner and standardized object-relational mappings.
- It adds support for polymorphism and inheritance.
- Supports *JPQL*, and improved version of *EJB QL*.
- Makes it easier to test entities outside of the EJB container.
- It can be used outside of the container.

It is because of these advantages, JPA specification is chosen in this thesis work. Many Java developers prefer lightweight persistent objects, which is supported by open-source frameworks or data access objects instead of Entity Beans. This is because entity beans and enterprise beans were too heavyweight and complicated compared to light weight and simple persistent objects [35][36].

2.2.4 Hibernate

The contradiction between data model and logic model is the main problem in the traditional development process of J2EE applications. The database currently in use are basically relational database, but Java is essentially an object-oriented language, whose object uses SQL and JDBC for database operations in storing and reading, which reducing the efficiency of programming and maintainability of the systems. The reality is that, if we write the SQL code manually in the enterprise applications, and it will cost a lot of development time to update and maintain the persistent layer. It might be nice to save the Java objects into the relational database. The system which adopts J2EE platform and selects Hibernate to design persistence layer can improve reusability of the system underlying business logic and also increases the system scalability [37][38]. Figure 6 shows that Hibernate provides persistence services and persistent objects to the application layer by using the database and configuration data file.

After extensive review, in this work the J2EE application development strategy is implemented based on JPA and Hibernate framework, in order to reduce the code coupling and increase the efficiency of system development.

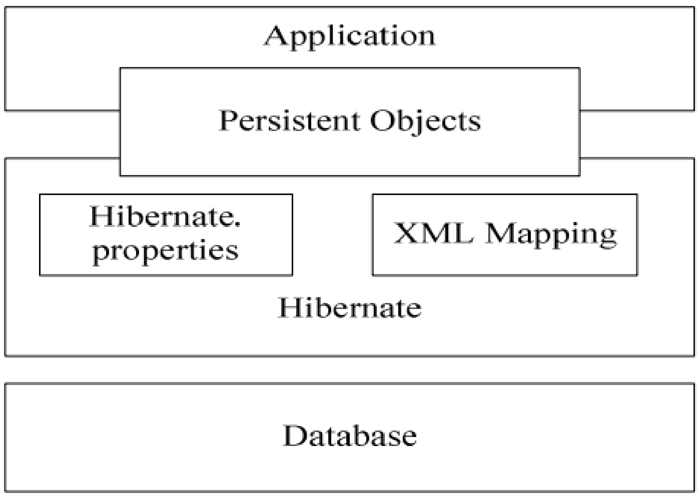


Figure 6. *The architecture of hibernate framework [37].*

2.3 Database Modeling in UML

UML[5] stands for Unified Modeling Language. It was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. It represents a unification of the concepts and notations presented by the three authors in their respective books [39][40][41]. The goal is for UML to become a defacto standard modeling language for visualizing, specifying, constructing and docu-

[5] UML, Available online: http://www.uml.org/

menting the object oriented software systems, as well as for business modeling and other non-software systems.

Although UML is mainly used for modeling software systems but it also can be used with all processes throughout the development lifecycle and across different implementation technologies. In order to model in UML, one has to obtain a compliant modeling tool from one of these providers.

2.3.1 Modeling Types

A conceptual model is the primary requirement for creating an UML diagram. Conceptual model is defined as a model which is made up of concepts and relationships. Concepts are real world entities and relationships are the methods in which the real word entities interact with each other.

A conceptual model of UML can be described by following three main elements:

- *UML building blocks*
- *Rules to connect the building blocks*
- *UML mechanisms*

UML offers a variety of diagrammatic notations for modeling static and dynamic aspects of an application. UML plays an important role in defining different perspectives of a system. These perspectives are:

- ***Design*** of a system consists of classes, interfaces and collaboration. UML provides class diagram, object diagram to support this.
- ***Implementation*** defines the components assembled together to make a complete physical system. UML component diagram is used to support implementation perspective.
- ***Process*** defines the flow of the system. So the same elements as used in *Design* are also used to support this perspective.
- ***Deployment*** represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective.

Based on these perspective, UML modeling can be classified into two types: structural and behavior modeling [43]. The structural modeling describes the static features of the system and its parts on different abstraction and implementation levels and how they are related to each other. And the behavior modeling shows the dynamic features of the objects in a system, which can be described as a series of change to the system over the time. The Figure 7 elaborates the types of diagram in UML 2.4 version.

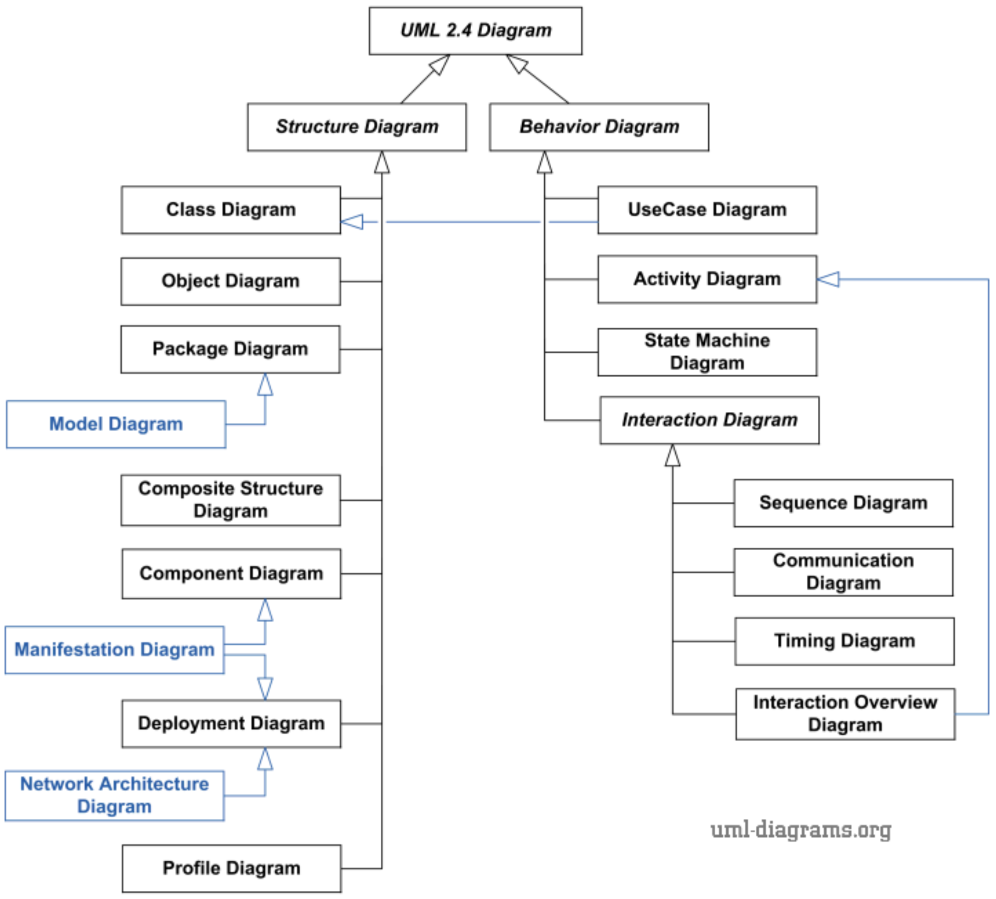


Figure 7. *Diagrams in UML 2.4 version [43].*

2.3.2 Modeling UML diagrams

The five fundamental UML models are use case, class, sequence, state and activity diagrams. In this thesis, class diagram, use case, sequence and package diagrams are used to model the system.

Class Diagram

A class diagram is used to depict the classes within a model to produce a logical structure of a software system. It is used both in analysis and the design phases. The class diagram emphasizes how the different entities related to each other. In other words, it shows a collection of classes, interfaces, associations, collaborations and constraints. Hence it is also called as structural diagram. The Figure 8 describes the major elements present in a class diagram.

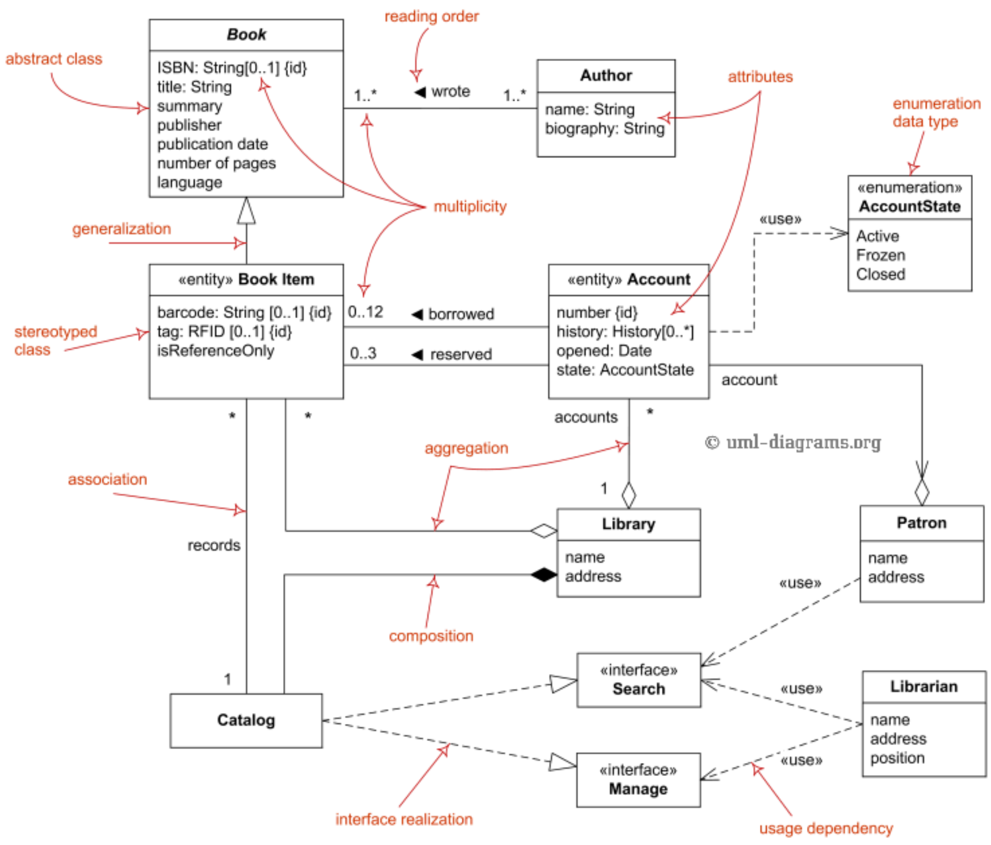


Figure 8. *The major elements of class diagram [43].*

Interaction Diagrams

The *Interaction Diagrams* evidently emphasize the interactions among the various elements in the software model [43]. There are two types of interaction diagram in UML which are used to represent the interactions, namely the sequence diagram and the communication diagram. Even though both diagrams are used to visualize the exchange of information, the emphasis is different.

Communication diagrams emphasize the relationships of individual objects and their topology. *Sequence diagrams* emphasize the chronological sequence of exchanged information. Hence both the diagrams are used to illustrate the dynamic behavior of the system form different perspective. The Figure 9 shows the majors elements of a sequence diagram.

In this work, the representations were made through sequence diagrams and without communication diagrams for two reasons:

1. Developers and readers understand the sequence diagrams easily than collaboration diagrams. Since sequence diagrams are simpler in most of the projects readers tend to show much higher acceptance of sequence diagrams.

2. In order to avoid decrease the number of unnecessary diagrams to represent the same idea.

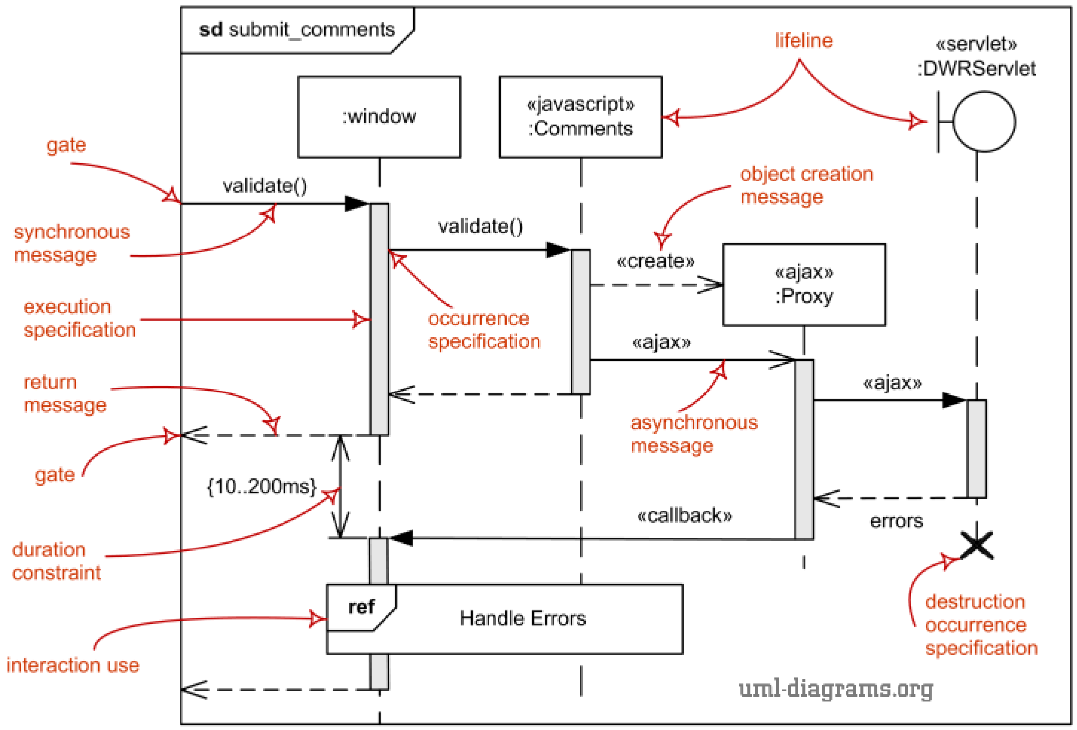
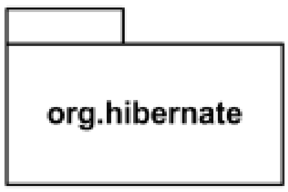


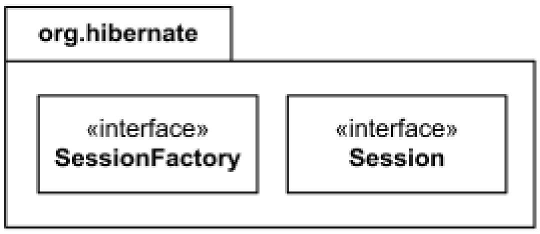
Figure 9. *The major elements of sequence diagram [43]*

Package Diagram

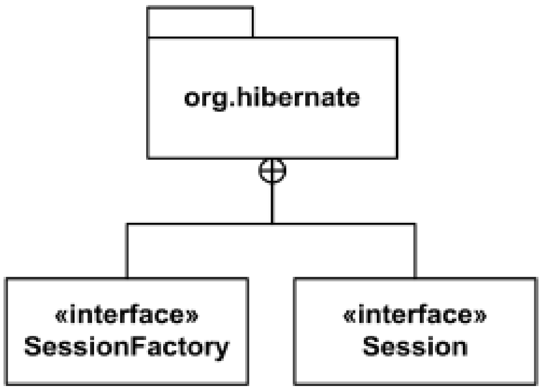
A package diagram is UML structure diagram which depicts the dependencies between different packages in a software model [43]. The package diagrams can be represented in different ways depending upon content of the package. For example, the Figure 10a) represents a package without any members, the Figure 10b) illustrates a package diagram with complete access of members and finally Figure 10c) describes a package diagram which only allows to access the subset of the contained elements.



a) Package diagram of org.hibernate.



b) Package diagram of org.hibernate, which contains Session Factory and Session.



c) Package diagram of org.hibernate, which contains subsets Session Factory and Session.

Figure 10. *Different methods of packaging [43].*

And also there are different types of UML dependency relationships. Among them the three main dependency relationships are *element import*, *package import* and *package merge* [43].

2.4 Data Access Object and Persistence Layer

The section 2.4.1 describes the design methodology for Data Access Object (DAO). And the section 2.4.2 demonstrates the concepts of Persistence Layer with an example.

2.4.1 Data Access Object

Many real world applications consist of business objects, which collaborate in order to perform some needed business functionality. Some of this business objects need to persist their data at some point. A typical implementation of such a business object is shown in Figure 11. As the figure shows how the data access logic is embedded into the business object, so the business object not only knows its data but it also has the knowledge of how and where to access it [46].

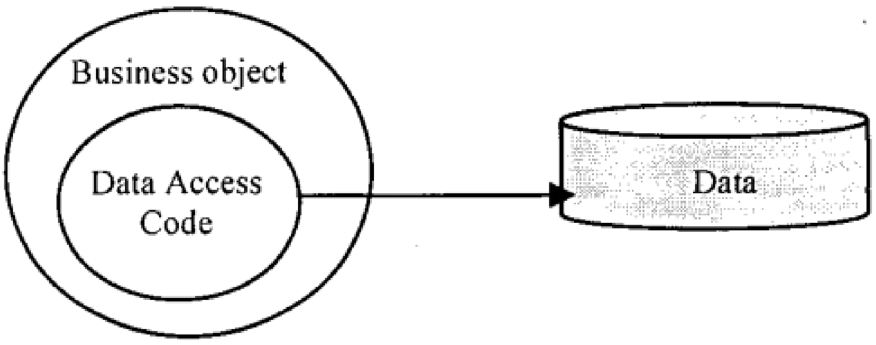


Figure 11. *Business object, which encapsulates data access logic [46].*

In order to access the data from a repository there are different database connectivity mechanisms. The mechanisms help the application programs to connect and communication with repository and it varies depending on data repository. These database connectivity mechanisms are also known as database middleware [47].

A Data Access Object (DAO) patterns provide an abstraction to a data source, such as a database [48]. The goal of this pattern is to provide the abstraction between business logic layer and persistent storage layer through encapsulating all the access to data resources and wrapping up the logical implementation details of data resources. Hence both business logic and persistent storage parts can evolve and be changed independently [49]. Figure 12 shows a class diagram representing the generic DAO pattern and Figure 13 represents the sequence diagram, which shows the interaction between the various objects in the pattern.

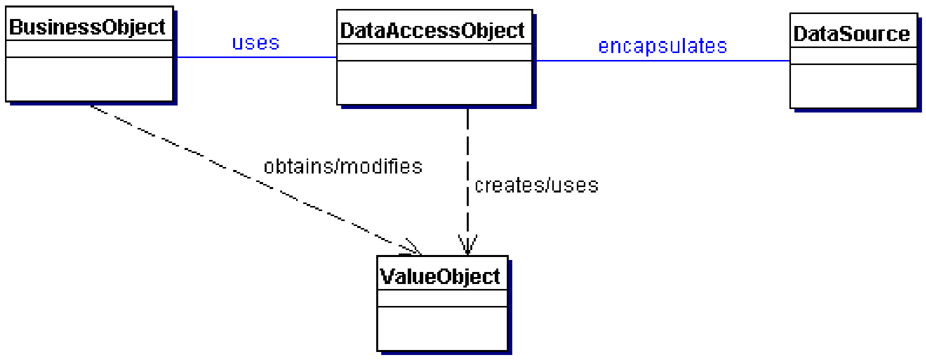


Figure 12. *Class diagram of DAO Pattern [48].*

The DAO pattern defines the following classes:

- ***BusinessObject*** represents the data client, which requires access to the data source to obtain, modify and store data. A BusinessObject may be implemented as a session bean, entity bean, or some other Java object, in addition to a servlet or helper bean that accesses the data source.

- ***DataAccessObject*** represents the primary object of this pattern. It abstracts the underlying data access implementation for the BusinessObject to enable transparent access to the data source. The BusinessObject also delegates data load and store operations to the DAO.
- ***DataSource*** represents a data source implementation. A data source could be a database such as an RDBMS, OODBMS, XML repository, flat file system, etc.
- ***ValueObject (TransferObject)*** used as a data carrier object. The DataAccessObject may use a ValueObject to return data to the client and may also receive the data from the client in a ValueObject to update the data in the data source.

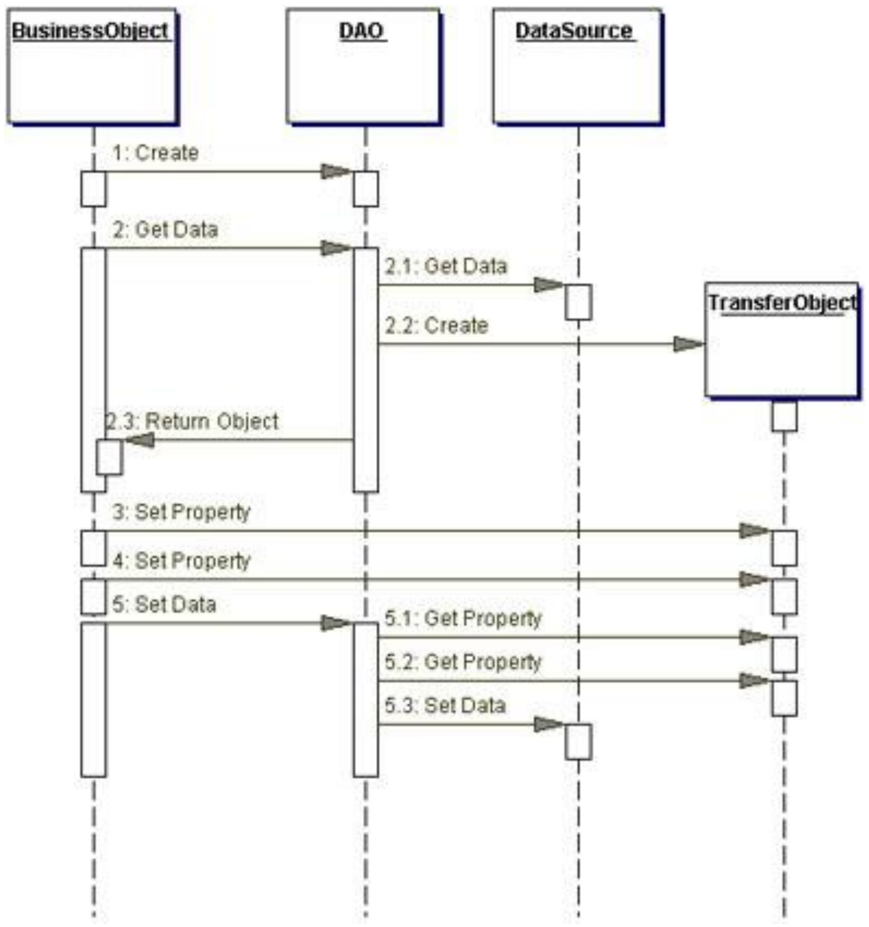


Figure 13. *Data Access Object sequence diagram [48].*

Access to persistent storage, such as to a database, varies greatly depending on the type of storage (relational databases, object-oriented databases, flat files, and so forth) and the vendor implementation.

2.4.2 Persistence Layer Concepts

Object persistence means that individual objects can outlive the process that created it. They can be saved to a data store and be re-created at the later point in time. The purpose of the persistence layer is to store retrieve and maintain entity domain components in a relational database. The approach to managing persistent data is a key design decision in almost every software application. The following figure describes the typical *Layered Architecture* that a software developer should follow while developing an application.

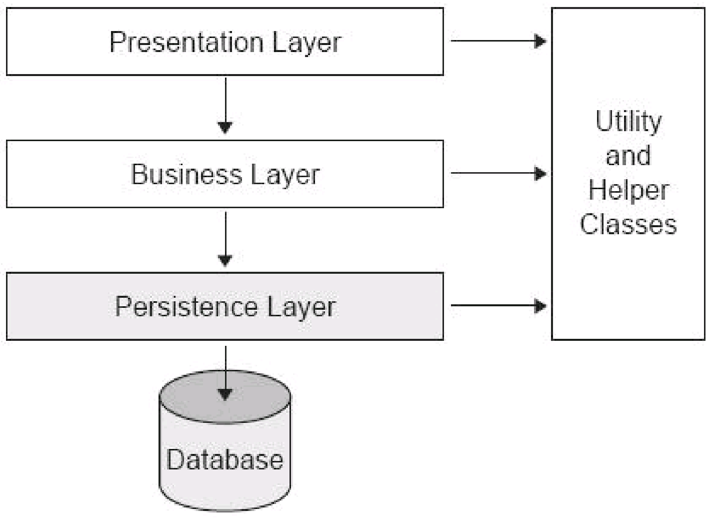


Figure 14. *Layered type architecture.*

Figure 14 indicates that users of the application interact directly with the presentation layer of the application. The application layer is generally made up of classes that implements visualization. Presentation classes are allowed to send messages to classes within the business (domain) layer and the utility (system) layer. The business layer implements the business classes of the application, for example the business layer for a robot manufacturing company would include classes such as *Customer* and *Robots*. The controller (process) layer, on the other hand, implements business logic that involves collaborating with several business classes or even other process classes such as the calculation of cost of robots (which would interact with instances of *Robots*, *Customer*, and *TypeOfRobots*). The system layer implements classes that provide access to operating system functionality such as printing and mailing electronically. Business classes are allowed to send messages to classes within the system layer and the persistence layer. The persistence layer encapsulates the behavior needed to store objects in persistence mechanisms such as object databases, files, and relational databases.

Figure 14 also describes that for the user-interface layer to obtain information it must interact with objects in the business layer, which in turn interact with the persistence layer to obtain the objects stored in your persistence mechanisms. This is an important feature of the layered type architecture – by not allowing the user interface of your application to directly access information stored in your persistence mechanism you effectively de-couple the user interface from the persistence schema. The implication is that you are now in a position to change the way that objects are stored, perhaps you want to reorganize the tables of a relational database or port from the persistence mechanism of one vendor to that of another, without having to rewrite your screens and reports.

The persistence layer can also takes care the following:

- Finding dependent object groups
- Managing application object identity
- Managing persistent object identities
- Persisting each object in the appropriate order
- Providing cache management
- Providing the proper transactional context
- Providing user-selectable locking modes

3. APPROACH

This chapter describes the existing architecture for mobility and ITS infrastructure in Tampere city and tools and technologies used to create new multi-modal architecture for Tampere region.

3.1 Existing mobility & ITS infrastructure in Tampere

Public mobility system in Tampere region is organized jointly between eight municipalities, Tampere, Pirkkala, Nokia, Kangasala, Lempäälä, Ylöjärvi, Vesijärvi and Orivesi. They provide the following Sub-Services.

3.1.1 Journey Planner for Cycling Sub-Service

This sub-service aims to provide the journey planner for cycling routes within Tampere region and national cycling touring routes. In order to use this sub-service, the user has to choose his departure and destination points. The sub-service also provides the locations for bike parking spots which are categorized into four types secure bike parking, bike parking, city-bike parking and bike-park and ride. The following Figure 15 shows the complete features of Tampere Cycling Journey Planner.

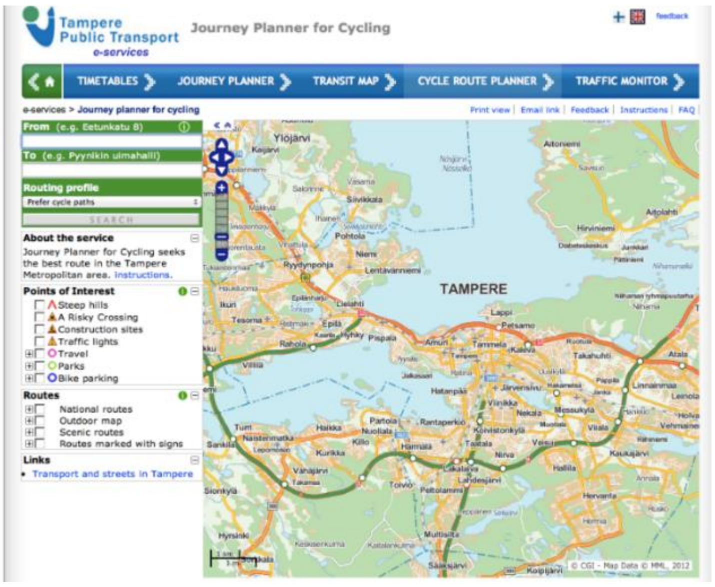


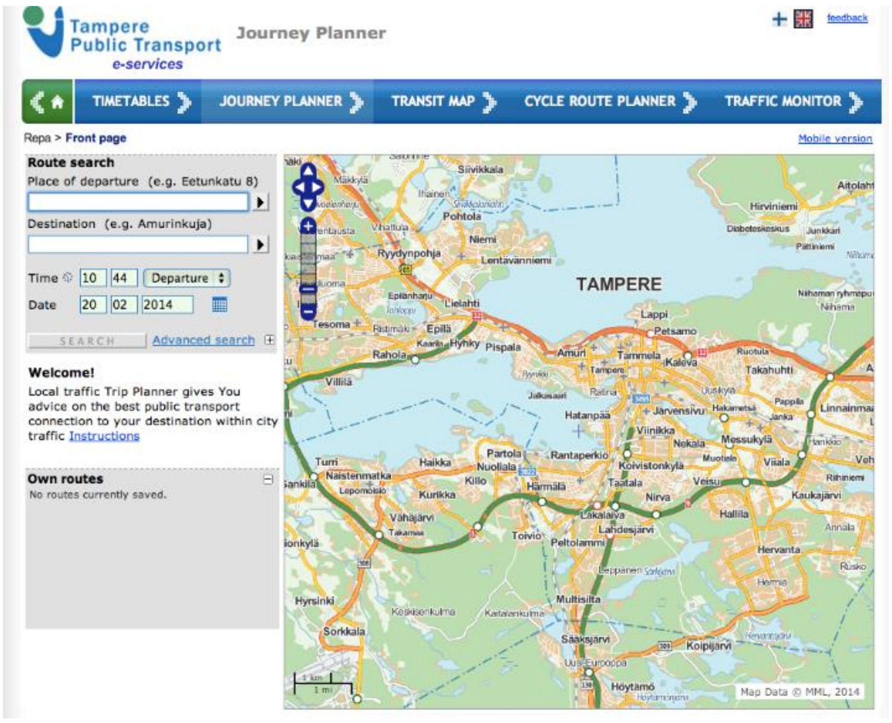
Figure 15. Tampere Cycling Journey Planner[6].

[6] Tampere Public Transport, Available online: http://kevytliikenne.tampere.fi/

3.1.2 Journey Planner for Buses Sub-Service

This sub-service aims to provide the journey planner for travelling in public buses within Tampere region. The Figure 16 illustrates the complete features of Tampere Bus Journey Planner.

The user has to enter the place of departure and destination along with time and date which is optional to avail this sub-service. It also has other distinctive additional feathers which are described according to user preferences. These features are walking speed of the user, maximum walking length the user wish to walk, type of route (fasters, least transfers, least walking) and the time limit for transfer safety margin.



Repa > Front page

Route search
Place of departure (e.g. Eetunkatu 8)
Destination (e.g. Amurinkuja)
Time 10 44 Departure
Date 20 02 2014

Welcome!
Local traffic Trip Planner gives You advice on the best public transport connection to your destination within city traffic Instructions

Own routes
No routes currently saved.

Figure 16. Tampere Bus Journey Planner[7].

3.1.3 Smart Traffic Prioritization Sub-Service

Smart Traffic Prioritization Sub-Service uses On-Board Units (OBUs) in buses send information such as GPS coordinates, line number and bus direction to a backend system at a frequency of one second. With these data the backend system controls real-time timetable monitors, traffic light priorities and data related to traffic disturbances. The Figure 17 describes the OBUs role in smart traffic prioritization sub-service.

A Fare Collection Unit (FCU) is been connected to each OBUs which is aware of information concerning the bus line in use and the bus direction. Both FCU and OBU data are combined and sent to the backend where data is processed to obtain accurate timeta-

[7] Tampere Public Transport, Available online: http://reittiopas.tampere.fi/

ble estimations of each bus and later the will be broadcasted to real time timetable monitors and internet-services.

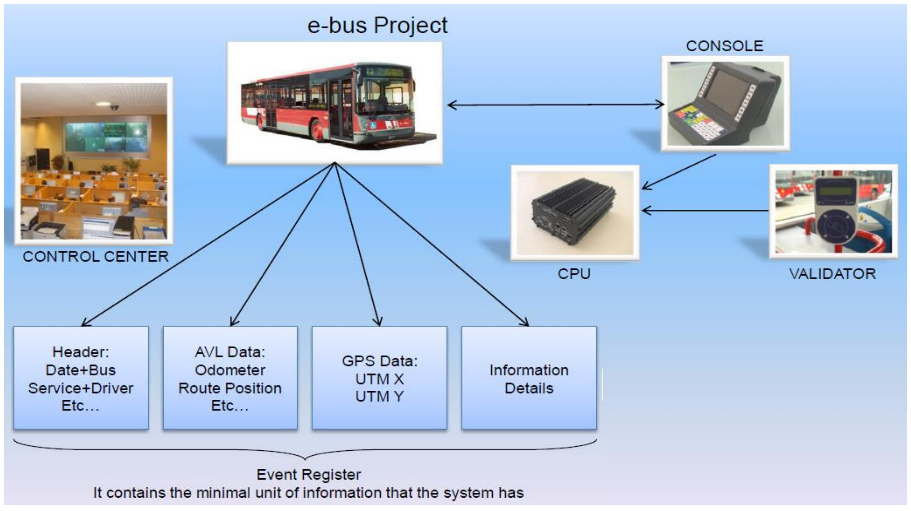
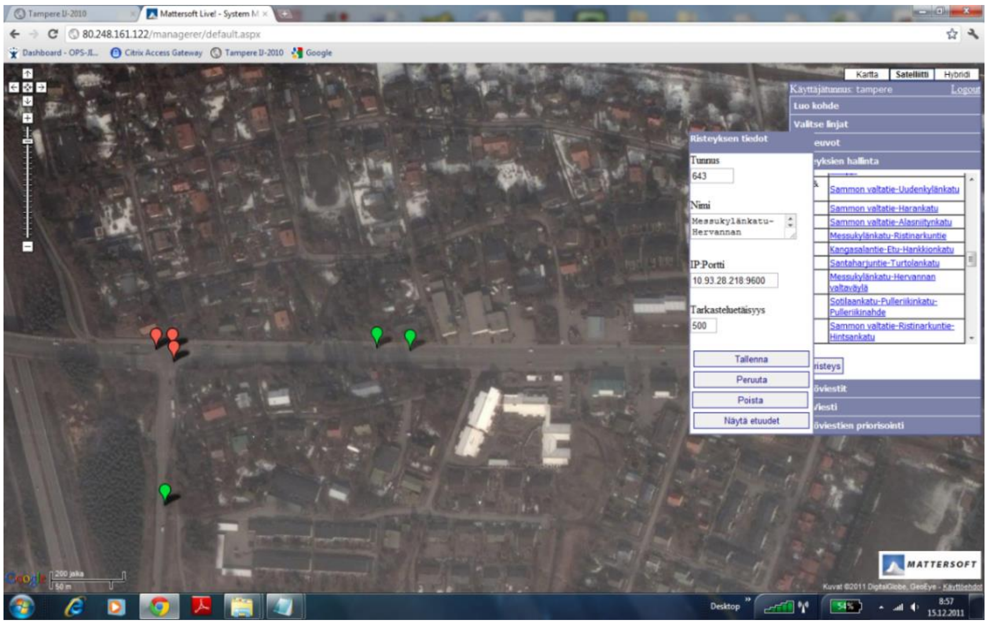

e-bus Project
CONSOLE
CONTROL CENTER
CPU
VALIDATOR
Header: Date+Bus Service+Driver Etc…
AVL Data: Odometer Route Position Etc…
GPS Data: UTM X UTM Y
Information Details
Event Register
It contains the minimal unit of information that the system has

Figure 17. *General Architecture of OBUs.*

With the help of existing system, the control center can possibly provide the following types of smart prioritization.

- Possibility to adjust traffic light prioritization upon request from the OBU of a vehicle, within second's margin to any specific crossing in case of bus is late according to the timetable and emergency vehicles such as fire trucks, ambulances and police with emergency lights on.
- Possibility to prioritize traffic disturbances from the backend during special events such as concerts.
- Traffic light priority system can be controlled also from backend by pinpointing virtual request and confirmation points on system's map. After bus crosses virtual request point, the system automatically sends traffic light priority request. After crossing confirmation point the priority is set off. Figure 18 shows the screen shot of virtual prioritization request from the control center.
- Four types of traffic light priorities can be provided with the existing system. And they are *Early Green Light*, *Extended Green Light*, *Extra Green Light Cycle* and *Cycle Change* in whole crossing if possible. These prioritization information is sent to the traffic light control unit with 3G-mobile connection.


Risteyksen tiedot

Tunnus: 643

Nimi: Messukylänkatu-Hervannan

IP:Portti: 10.93.28.218:9600

Tarkasteluetäisyys: 500

Käyttäjätunnus: tampere

80.248.161.122/managerer/default.aspx
Kartta
Satelliitti
Hybridi
Logout
Luo kohde
Valitse linjat
Tallenna
Peruuta
Poista
Näytä etuudet
Sammon valtatie-Uudenkylänkatu
Sammon valtatie-Harankatu
Sammon valtatie-Alasniitynkatu
Messukylänkatu-Ristinarkuntie
Kangasalantie-Etu-Hankkionkatu
Santaharjuntie-Turtolankatu
Messukylänkatu-Hervannan valtaväylä
Sotilaankatu-Pulleriikinkatu-Pulleriikinahde
Sammon valtatie-Ristinarkuntie-Hintsankatu
MATTERSOFT

Figure 18. *Virtual prioritization request [42].*

Based on the assessment of available infrastructure, a qualitative analysis is performed for the services described in this thesis work. The various ITS datasets which are grouped according to the reference ITS areas are as follows:

- *Traffic Management*
- *Emergency Notification and Response*
- *Public Transport Management*
- *Traveler Journey Assistance*
- *Electronic Fee Collection*
- *In-Vehicle Systems*
- *Law Enforcement.*
- *Communications*
- *Demography & Economics*
- *Tourism*
- *Businesses*
- *Crowd Sourced Data*

The Table 4 describes the qualitative evaluation of the availability and effectiveness of the above data sets in relation with each mobility services described in this thesis work. This analysis is measure with this scale:

Table 4. *Qualitative evaluation of datasets in Tampere.*

Services	Traffic Management	Emergency Notification and Response	Public Transport Management	Traveler Journey Assistance	Electronic Fee Collection	In-Vehicle Systems	Law Enforcement	Communications	Demography & Economics	Tourism	Business	Crowd sourced data
Multi-modal Journey Planner	6	5	5	6	NR	0	NA	0	NR	0	NA	0
Energy efficiency / CO_2 assessment of journey options[8]	*6*	5	*5*	NR	NR	0	NR	0	NR	0	NR	0
User tailored incentive-based visualization[5]	6	NR	*5*	6	NR	0	NA	0	NR	0	NR	0

This analysis is measure with this scale:

NR: datasets are *Not Relevant*
NA: not enough information on datasets
6: datasets are sufficient for the service and ready to be used without criticisms
5: datasets are sufficient for the service and ready to be used with some criticisms
4: datasets are sufficient for the service and ready to be used with many criticisms
3: datasets are partially sufficient for the service and ready to be used without criticisms
2: datasets are sufficient for the service and ready to be used with some criticisms
1: datasets are sufficient for the service and ready to be used with many criticisms
0: datasets are not sufficient for the service

[8] These services rely on the data generated by the users through the app registration which should contain attributes related to each services such as carpooling allowance, travelling route, mobility choice, fuel input etc.

For the Multi-modal Journey Planner service, *5* has been granted to the Public Transport because the available datasets are currently available to produce bus and bike planners. A Car Journey Planner must be developed as a last step towards achieving full integration, which is achieved in the implementation section of this thesis. And the service EE / CO_2 assessment of journey options relies indirectly (through the service Multi-modal Journey Planner) for the datasets of *TMS* and *Public Transport Management*. *5* is granted to the *Emergency Notification* datasets on the scale of average. At this moment, the maintenance vehicles collecting real-time winter road maintenance data are equipped with OBUs, but the actual collection of the datasets has not yet started. Also all other needed datasets such as road weather and frost heave are within the *Emergency Notification* datasets are awarded *6* in the evaluation. *Law Enforcement* datasets exist, but are in use by official authority only, not made public to all developers.

3.2 Tools and Framework

This section encompasses the tools and frameworks used in the implementation of this thesis work. These tools and framework are selected based on the detailed review performed in pervious chapter section 2.2.

3.2.1 ArgoUML

ArgoUML[9] is a pure-Java based open source Unified Modeling Language (UML) modeling tool based on all standard *UML 2.4 diagrams* specifications [43]. This tool used for developing UML model and stating that model with graphical artifacts as specified by the UML standard. It uses XML formats such as XMI and PGML. The Figure 19 is a screen shot of *ArgoUML* tool, taken while developing the class diagram for this thesis work.

It provides a unified, consistent way to communicate information about software systems in a way designed to be intuitive for both technical and non-technical individuals. It is distributed under the *Eclipse Public License (EPL) 1.0*[10]. For this thesis work, *ArgoUML 0.34* was used to implement the commonly encountered UML artifacts.

[9] ArgoUML, Available online: http://argouml.tigris.org/
[10] Eclipse Public License, Available online: https://www.eclipse.org/legal/epl-v10.html

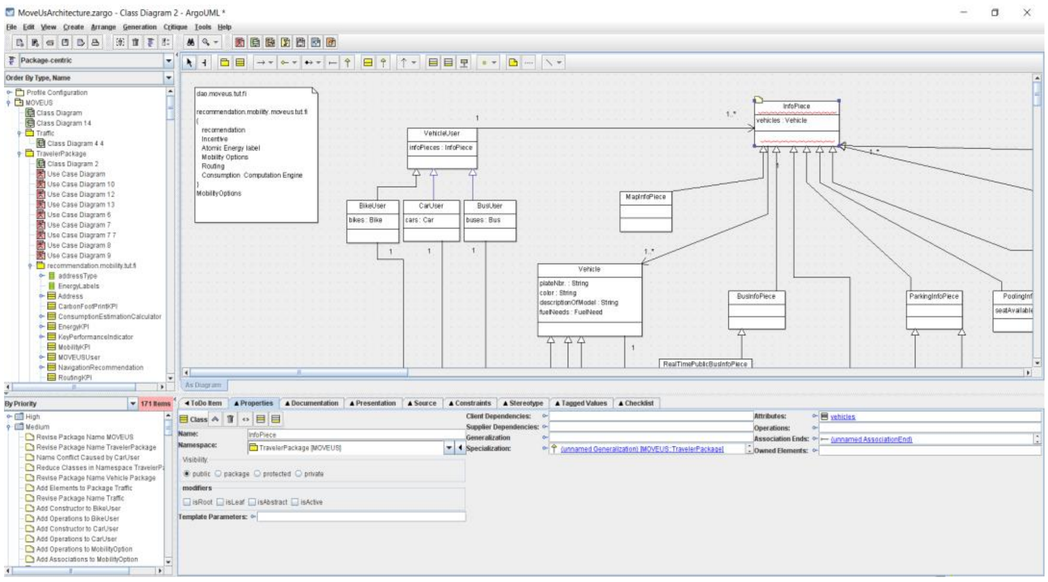


Figure 19. *Screen shot of ArgoUML.*

3.2.2 Java

Java[11] is concurrent, class-based and object oriented programming language. During this entire thesis work, Eclipse[12] (Kepler Service Release 1) IDE is used in the implementation. Java SE 7 (Update 51) and JRE7 are used throughout in the implementation process. It is platform independent language. Java programs are compiled to an intermediate representation called byte-code that can run on any Java Virtual Machine (JVM) nevertheless on any type of computer architecture.

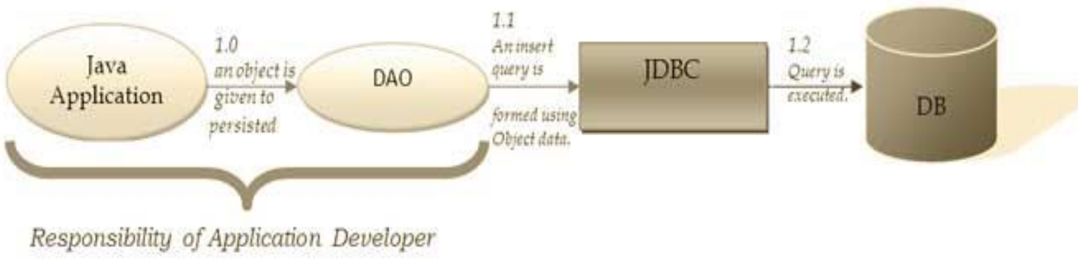
3.2.3 Hibernate Object Related Mapping

Hibernate Object Related Mapping[13] (ORM) is a high performance object-relational mapping framework which is licensed under the open source GNU Lesser General Public License (LGPL). During this work, Hibernate 4.3.1 version is used. It provides a framework for mapping an object-oriented domain model to a relational database. It also provides API's, which are implementation of the *Java Persistence API (JPA) specification*[14].It uses various other existing Java APIs such as Java Database Connectivity (JDBC), Java Transaction API (JTA), and Java Naming and Directory Interface (JNDI).

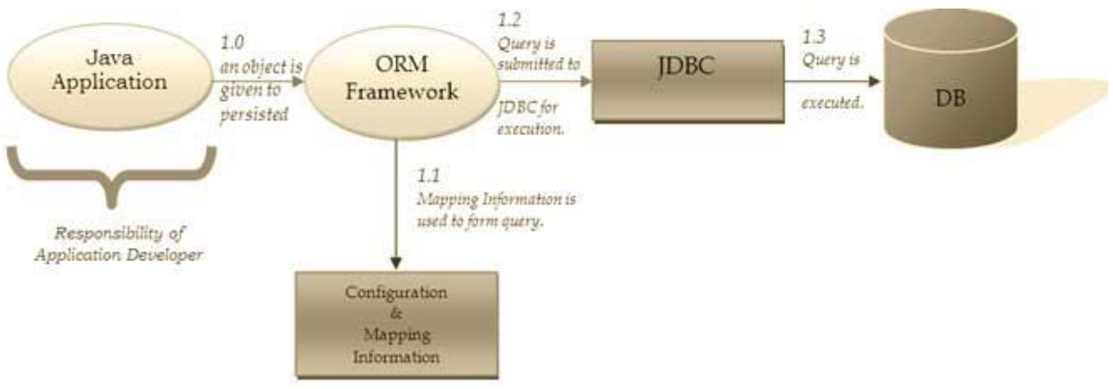
JDBC provides a rudimentary level of abstraction of functionality common to relational databases, allowing almost any database with a JDBC driver to be supported by Hibernate. JNDI and JTA allow Hibernate to be integrated with J2EE application servers.

[11] Java, Available online: https://www.oracle.com/java/index.html
[12] Eclipse IDE, Available online: http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/marsr
[13] Hibernate, Available online: http://hibernate.org/orm/
[14] JSR-317:Java Persistence 2.0 , Available online: https://jcp.org/en/jsr/detail?id=317

In this work, hibernate helps in achieving persistence. Persistence is the term use to describe the application data to outlive the application process. It allows the data to stay in the same state beyond the scope of the JVM. The Figure 20a) and Figure 20b) describes the difference between with and without persistence.


Java Application
1.0 an object is given to persisted
DAO
1.1 An insert query is formed using Object data.
JDBC
1.2 Query is executed.
DB
Responsibility of Application Developer

a) Persistence of an Object without ORM


Java Application
1.0 an object is given to persisted
ORM Framework
1.2 Query is submitted to JDBC for execution.
JDBC
1.3 Query is executed.
DB
1.1 Mapping Information is used to form query.
Configuration & Mapping Information
Responsibility of Application Developer

b) Persistence of an Object using ORM.

Figure 20. *Persistence of an object with and without ORM*[15].

3.2.4 JUnit

A Unit Test Case is a part of code which ensures the method works as expected. To achieve those desired results quickly, test framework is required. JUnit[16] is a unit testing framework to write repeatable tests. It has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit that originated with JUnit. The Figure 21 describes the flow chart of a software unit testing life cycle.

[15] Hibernate 3.1 Tutorial, Available Online: http://www.jobsacid.com/java-tutorial/advanced-java/hibernate/hibernate-introduction.html
[16] JUnit, Available Online: http://junit.org/junit4/

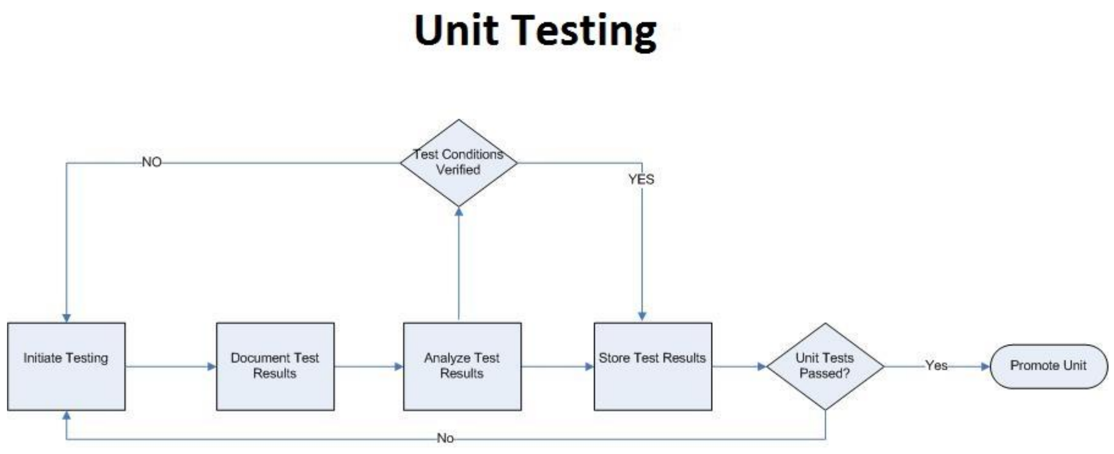


Figure 21. *Flow chat of Unit Testing cycle*[17].

3.2.5 Apache Derby

Apache Derby[18] or simply Derby is an open source database implemented purely in Java. It is transactional, embeddable and pure-Java relational database based on JDBC and SQL standards. It’s available under the *Apache License Version 2.0*.The Figure 22 illustrates the embedded engine architecture of Derby.

Some advantages of derby are it has embedded JDBC driver that allows the developer to embed derby in any Java-based solution and it’s easy to install, deploy and use [44]. *Apache Derby version 10.10.1.1* database is used in this work.

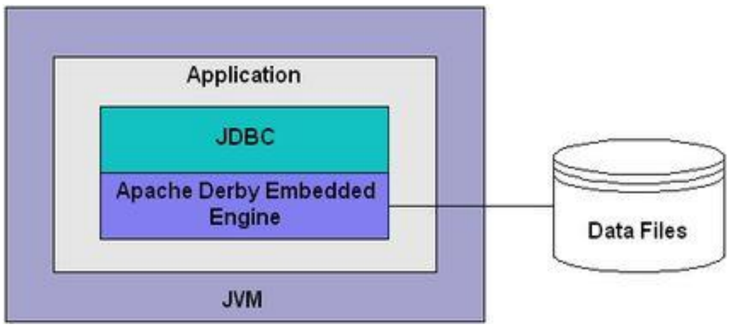


Figure 22. *Embedded engine architecture of Apache Derby.*

[17] Software Testing, Available online: https://www.techaheadcorp.com/wp-content/uploads/2015/08/Unit-Test-Execution.jpg
[18] Apache Derby, Available online: https://db.apache.org/derby/

3.2.6 Apache Tomcat

Apache Tomcat[19] or simply Tomcat, is an open source software developed by Apache Software Foundation (ASF). It acts as web server and servlet container with the implementation of Java Servlet, Java Server Pages, Java Web Socket technologies and Java Expression Language. In this thesis work Tomcat v7.0 is used as a web server.

3.2.7 Intelligent Transport Systems and Services - Factory

Intelligent Transport Systems and Services (ITS) provides wide range of data from traffic data via data processing unit from various sector and these data are used to produce traffic and transportation related services.

Data are collected from sensors installed in vehicles, from weather stations, from traffic cameras, and so on. And these data can be used to provide services such as Traffic monitoring and control, private car navigation, public transportation, and so on.

The city of Tampere provides make several traffic data sets open for both private and commercial purpose, in order to create new products and services. This information is pertaining to Tampere section and all the APIs are real-time JSON. The Figure 23 illustrates the types of raw data feeds and data interfaces available in ITS-factory application [45]. The various information from ITS-factory was used to develop the object model for *MoveUs* application.

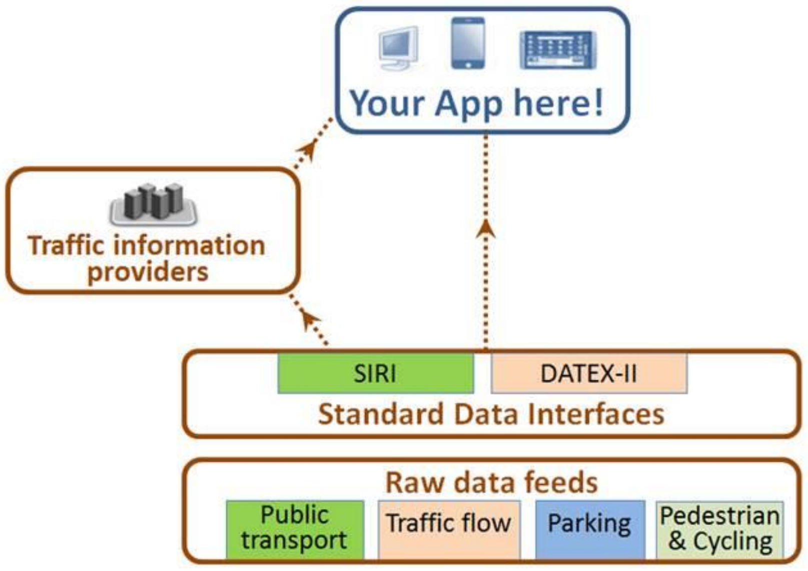


Figure 23. *ITS-factory Application*[20].

[19] Apache Tomcat, Available online: http://tomcat.apache.org/
[20] ITS-factory, Available on: http://wiki.itsfactory.fi/images/0/01/Its_factory_app.jpg

3.2.8 General Transit Feed Specification

General Transit Feed Specification[21] (GTFS) which defines a standard format for public transportation schedules and associated geographic information. GTFS feeds allow public transit agencies to publish their transit data and developers to write applications that consume the data in an interoperable way. The Figure 24 describes an example UML diagram of the table in a common GTFS feed.

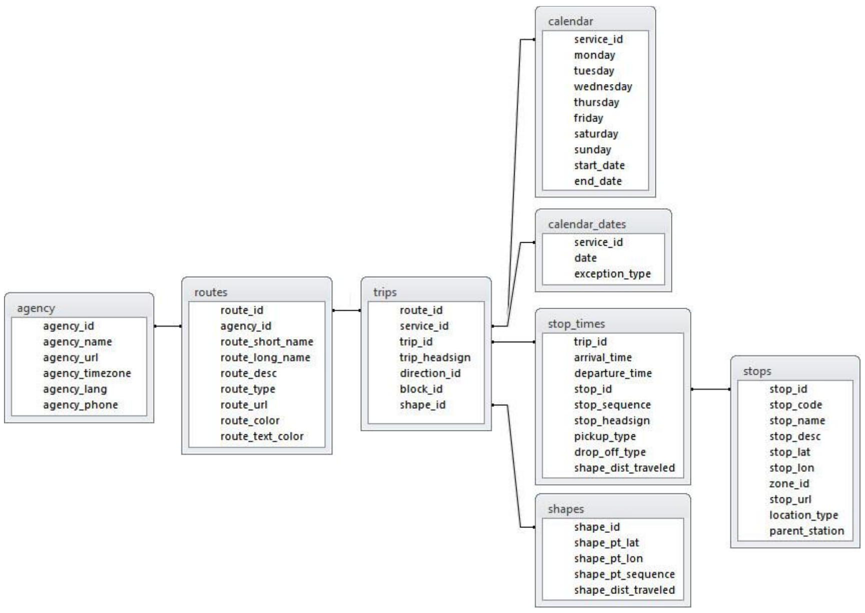


Figure 24. *An example UML diagram of the tables in a common GTFS feed*[22].

3.2.9 Extensible Markup Language

eXtensible Markup Language[23] (XML) is a structured language design to structure, store and transport data. It is a markup language such as Hyper Text Transfer Protocol (HTTP). It is defined by the W3C's XML 1.0 and several other specifications, all of which are free open standards. In this work, *"XML 1.0"* version is used for hibernate configuration and to define the persistence class. For editing the XML, EditiX[24] XML editor is used.

3.3 Tampere Mobility System

In this thesis work, the service structure for Tampere will depend mainly on the information from ITS Factory – a new innovation, experimentation and development environment, where companies and individual developers can develop, test and productize

[21] GTFS, Available online: https://developers.google.com/transit/gtfs/
[22] MARTA Developer Resources: Available online: http://www.joederose.us/MARTA/Data/
[23] XML tutorial, Available online: http://www.w3schools.com/xml/
[24] EditX, Available online: http://www.editix.com/download.html

traffic solutions. Also, in the section 3.2.7 detailed description about ITS Factory is discussed. The solutions can be built on top of a continuously updated base of traffic open data[25]. The ITS Factory supports the partners by guaranteeing the sustainability of the *MoveUs* applications in Tampere basically with following six categories:

- ***Test field:*** Maintenance of a network of traffic sensors which provides a model of the TRE region traffic infrastructure as a testing platform.
- ***End-user experience*:** ITS Factory is an environment for testing services and systems so that they can be made to fit the needs of the end users. This enables partners to design tailor made services for citizens, businesses and the public sector.
- ***Standardization*:** ITS charts existing and future standards with the aim of publishing open data in a standardized format, which allows scaling and duplication.
- ***Road mapping*:** Keeping track of open interfaces and data and standardization.
- ***Open data and developer support:*** Allowing the development of new applications and services.
- ***Marketing and export support:*** Aiding in marketing the product and providing consistent support after deployment.

Also in this work, for Tampere pilot region the target users will be essentially the same as in for other *MoveUs* pilot sites Genoa and Madrid. And the target users are:

- Public transportation users
- Municipality of Tampere
- Mangers of public transportation services
- Engineers of public transportation services

3.3.1 Architecture and Concept

In order to develop a sustainable, interoperable, and scalable ITS, it requires better usage of existing infrastructures. And also the integration of information and communication technologies should be consistent and coherent. The proposed multi-modal journey planner considers various factors such as the mode of transportation, travel route and the travel time. These informations are driven from various other source of data such as weather, road conditions, parking information etc. Table 5 contains *Functional ITS Architecture* already existing in Tampere. The informations from these sources were used in this thesis work.

[25] Tampere City Strategy, Available online: http://www.tampere.fi/material/attachments/t/5m6pXm17U/Tampere_City_Strategy.pdf

Table 5. *Functional ITS Architecture in Tampere.*

Reference ITS Areas	**Systems Implementing the ITS Areas**
Traffic Management	**Digitraffic**[26] is a service offering real time and historical information and data about the traffic on the Finnish main roads. The service is provided by the Finnish Transport Agency, and it is addressed for organization developing information services or working with traffic management and planning. **Digiroad**[27] is a national database which contains precise and accurate data on the location of all roads and streets in Finland as well as their most important physical features (covering a total of 483,000 km or 300,000 miles). **TrafficNow,** Tampere real time traffic situation from streets. Data is available to pilot for free of charge. **Parking Data,** parking halls and on-street parking
Public Transport Management (Traveler Journey Assistance)	**Traffic Monitor**[28], monitoring of Tampere bus traffic in real time (every one second) and visualization of predicted bus arrival times on stops. Public Transport Data in available both static and real time Data.
Public Transport Management (Traveler Journey Assistance)	**City Navigator**[29] is routing services with turn-by-turn navigation, supporting cyclists and public transport passengers. **Journey Planner for Public Bus**[30] provides recommendations on the best public transport connection to your destination within city traffic. **Journey Planner for Cycling**[31], seeking the best cycling route to user-defined points of interest in the Tampere Metropolitan area.

[26] Digitraffic, Available Online: http://www.infotripla.fi/digitraffic/english/index.html
[27] Digiroad, Available Online: http://www.digiroad.fi/hyodyntaminen/en_GB/ordering/
[28] Traffic Monitor, Available Online: http://lissu.tampere.fi
[29] City Navigator, Available Online: http://dev.itsfactory.fi/citynav/
[30] Journey Planner for Public Bus, Available Online: http://reittiopas.tampere.fi

3.3.2 Description of services

After detailed research, it is evident that a *multi-modal service* which combines the user with multiple based ITS architecture can solve the objective of this thesis work. However, the objective cannot be solved just with a *multi-modal service*, an additional service which increases the number of users with strong business layer connection is required. This requirement is met with the help of *User-tailored incentive-based visualization service.* At the backend of this service, incentives are created based on the selection of the mobility and routing options. In the later part of this chapter contains discussion about the UML design for these services, it also demonstrates the relation and communication between the proposed services and finally, fusion of existing ITS architecture to reform a *multi-modal service*.

The later part of this section contains a brief description of services which are implemented in this thesis work.

Multi-modal Journey Options

This service aims to offer the user the possibility to see all available mobility (i.e. bus/car/bike/pedestrian) and routing (streets and pathways) options between its current location and a declared intended destination.

Energy efficiency /* CO_2 *assessment of journey options

This service aims to assess the energy efficiency and / or CO_2 cost of input journey (i.e. mobility & routing) options between a source and a destination point, per user.

User-tailored incentive-based visualization service

This service aims to give an incentive oriented view of input transportation options information, per user. It acts as an adaptor for user-friendly meaningful display of backend computed information.

In *MoveUs*, for the Tampere pilot, these service targets the mapping of EE/CO_2 labels to incentive points (and subsequently relevant incentives) for the user who consider the journey options recommended by *MoveUs* application. And these incentives are based on a *Set of Incentive Rules* stored in a dedicated database.

[31] Journey Planner for Cycling, Available Online: http://kevytliikenne.tampere.fi

3.4 System Design

This sections describes the system design using UML diagrams. In this thesis work, use case diagrams and sequence diagrams are used to model the proposed services for *MoveUs* application. The section 3.4.1 demonstrates the use cases used for implementing the services. In 3.4.2 section, the sequence diagrams are explained in detail.

3.4.1 Use Case Diagrams

3.4.1.1 User – Application

This use case Figure 25 illustrates the interactions between the user and the *MoveUs* App. The user can enter his information about his current location and journey endpoints information. The user can visualize his journey options and energy consumption information.

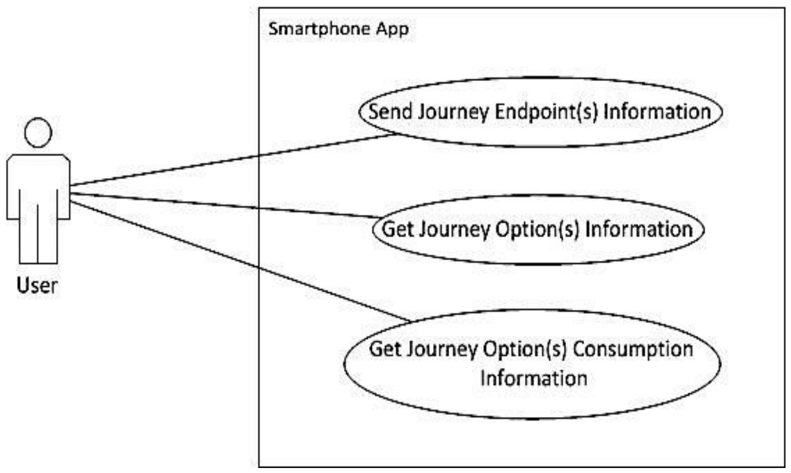

Smartphone App
User
Send Journey Endpoint(s) Information
Get Journey Option(s) Information
Get Journey Option(s) Consumption Information

Figure 25. *Use case diagram between User and Application.*

3.4.1.2 User – MoveUs Platform

This use case Figure 26 illustrates the interaction between the user and *MoveUs* Platform. The *MoveUs* platform allows the user to download the *MoveUs* application. The user can update his profile information and access his personal information. The journey ends point are the main inputs in this use case. The *MoveUs* platform provides the multiple journey options based on the user input. The user can visualize the selected journey option with incentive-based map.

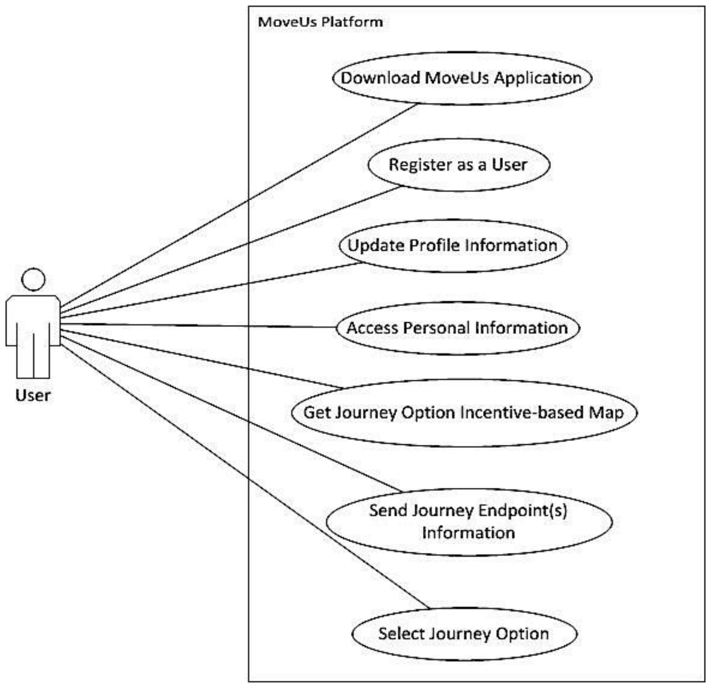

MoveUs Platform
Download MoveUs Application
Register as a User
Update Profile Information
Access Personal Information
Get Journey Option Incentive-based Map
Send Journey Endpoint(s) Information
Select Journey Option
User

Figure 26. *Use case diagram between User and MoveUs Platform.*

3.4.1.3 Application – MoveUs Platform

This use case Figure 27 illustrates the interaction between the application and the *MoveUs* Platform. When the user requests for consumption values the application sends the journey options to the *MoveUs* platform and it sends the consumption values.

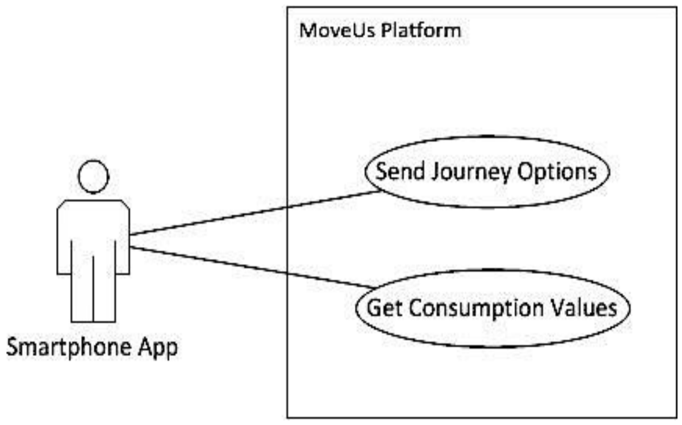


Figure 27. *Use case diagram between Application and MoveUs Platform.*

3.4.1.4 MJP service – Sub-services

This use case Figure 28 describes the interaction between the MJP service and the sub-services. MJP service comprises of three main sub-services such as the car journey planner, bus journey planner and bike journey planner. The main service can interact with these sub-service when the user request for a journey planner.

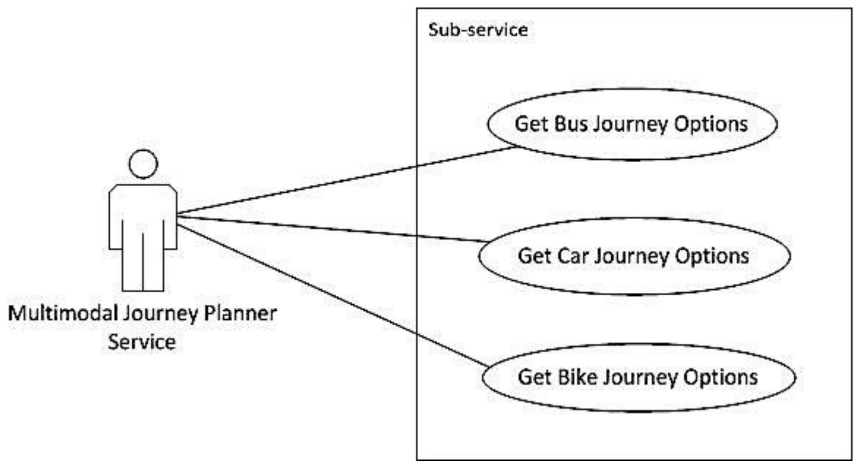


Figure 28. *Use case diagram between MJP service and sub-service.*

3.4.1.5 Consumption Estimation Calculator service – Application

This use case Figure 29 describes the interaction between the Consumption Estimation Calculator service and the application. The service provides journey options with consumption information for the journey options selected by the user.

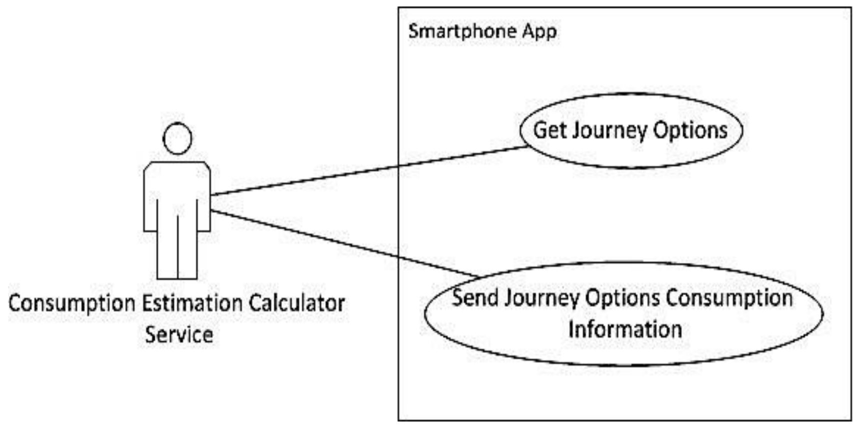

Smartphone App
Get Journey Options
Send Journey Options Consumption Information
Consumption Estimation Calculator Service

Figure 29. *Use case diagram between Consumption Estimation Calculator Service and Application.*

3.4.2 Sequence Diagrams

3.4.2.1 Calculation of Multi-modal Journey Options

This service aims to provide users with journey via bike, bus and car options with the smartest route between their actual position and a selected destination. In order to avail this service, the user has to enter his current location, his mobility preferences and the chosen destination. The output of this service is to show the smartest way to reach the destination taking into consideration all available mobility options, traffic situation, weather station data and routing options. The architecture for this service is illustrated with the help of sequence diagram which is shown in the Figure 30.

Within the smartest options that this service offers, users can choose how they want to move: public transport, car-sharing, or bike-sharing. Once selected, the application will display the smartest way and will guide the user to achieve the destination point. The route proposed will include user's preferences and incentives.

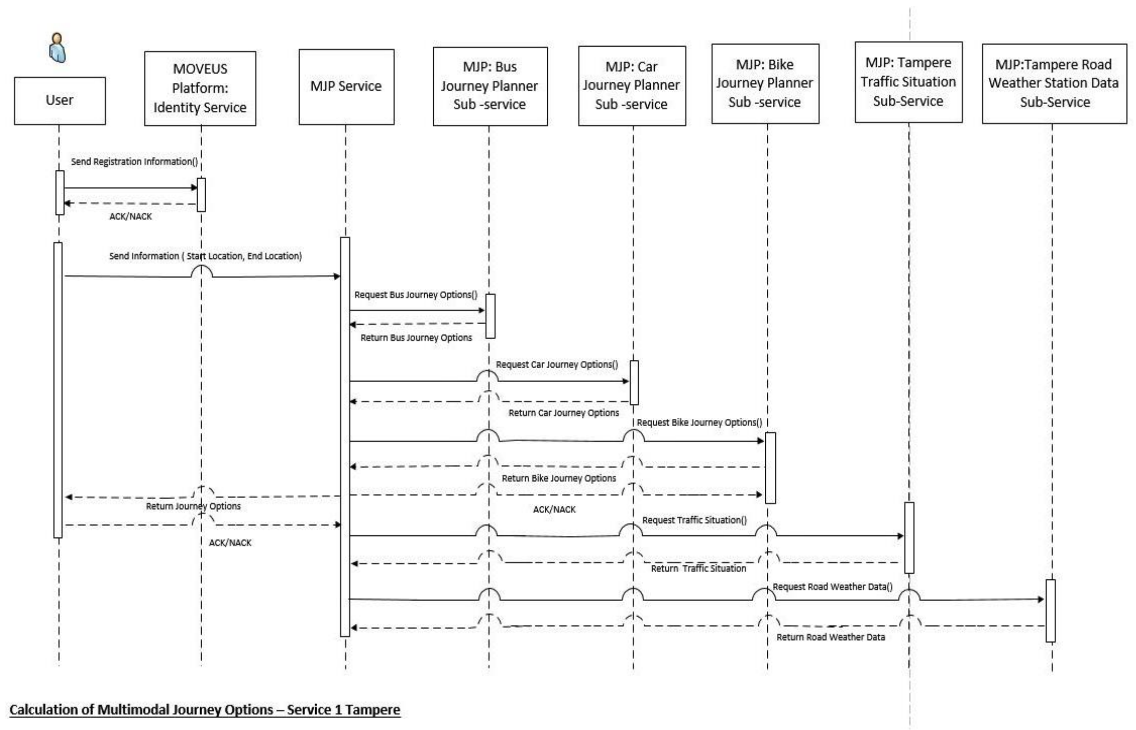

User
MOVEUS Platform: Identity Service
MJP Service
MJP: Bus Journey Planner Sub -service
MJP: Car Journey Planner Sub -service
MJP: Bike Journey Planner Sub -service
MJP: Tampere Traffic Situation Sub-Service
MJP:Tampere Road Weather Station Data Sub-Service
Send Registration Information()
ACK/NACK
Send Information (Start Location, End Location)
Request Bus Journey Options()
Return Bus Journey Options
Request Car Journey Options()
Return Car Journey Options
Request Bike Journey Options()
Return Bike Journey Options
Return Journey Options
ACK/NACK
ACK/NACK
Request Traffic Situation()
Return Traffic Situation
Request Road Weather Data()
Return Road Weather Data
Calculation of Multimodal Journey Options – Service 1 Tampere

Figure 30. *Sequence diagram of Calculation of Multi-modal Journey options.*

The following steps describes the flow of request and response of the sequence diagram in the Figure 30:

1. ***User → MoveUs platform***
   - Enter *MoveUs* web site
   - Enter username and password and start the registration; a preliminary email for confirmation will be sent to user to check email address
   - Enter information on user *(→2):*
     - Name, address, VAT
     - Special roles (e.g. tourist, business)
     - Direct influencers of personal mobility choices (e.g. passion for biking / running, personal acceptance of car/bike sharing)
     - Indirect influencers of personal mobility choices (e.g. preference to commute to work / school in groups)
     - Direct influencers of personal routing choices (e.g. aversion towards specific places in the city)
     - Habits (e.g. regular itineraries undertaken)
     - Particularities of personal vehicles the user is regularly driving (fuel needs, consumption, etc.)

2. ***MoveUs Platform: Identity Service***
   - Store information about user (profile and a history of selected Journey Option(s).
3. ***User→ Multi-modal Journey Planner service (Tampere)***
   - Enter information on START LOCATION and END LOCATION *(→4)*
4. ***Multi-modal Journey Planner service (Tampere)***
   - Computes Journey Option(s) based on the input information provided. *(→5)*
5. ***Multi-modal Journey Planner service (Tampere) → User***
   - Provide information on all available Journey Option(s) – bus, bike and car. *(→6)*
6. ***User***
   - Selects one of the available Journey Option(s). *(→2)*

3.4.2.2 Estimation of Consumption (CO_2 / Energy) per Journey Option

This service aims to assess the Energy Efficiency (EE) and / or CO2 cost of input for the journey options between the actual position and a selected destination, per user.

Each user profile describes a special role of his desire such as tourist. And his personal choice of mobility and routing. The *MoveUs* App user starts the *MoveUs* App and enters his desired destination and requests an assessment of the EE/CO_2 cost of the journey. The *MoveUs* EE/CO_2 assessment service identifies the Global Positioning Satellite (GPS) location of the requestor and the MJP service computes the possible mobility and routing options for the given source and destination point. The Consumption Estimation Calculator Service automatically computes the EE/CO_2 consumption associated to each identified journey option. The below Figure 31 illustrates the sequence diagram for this service.

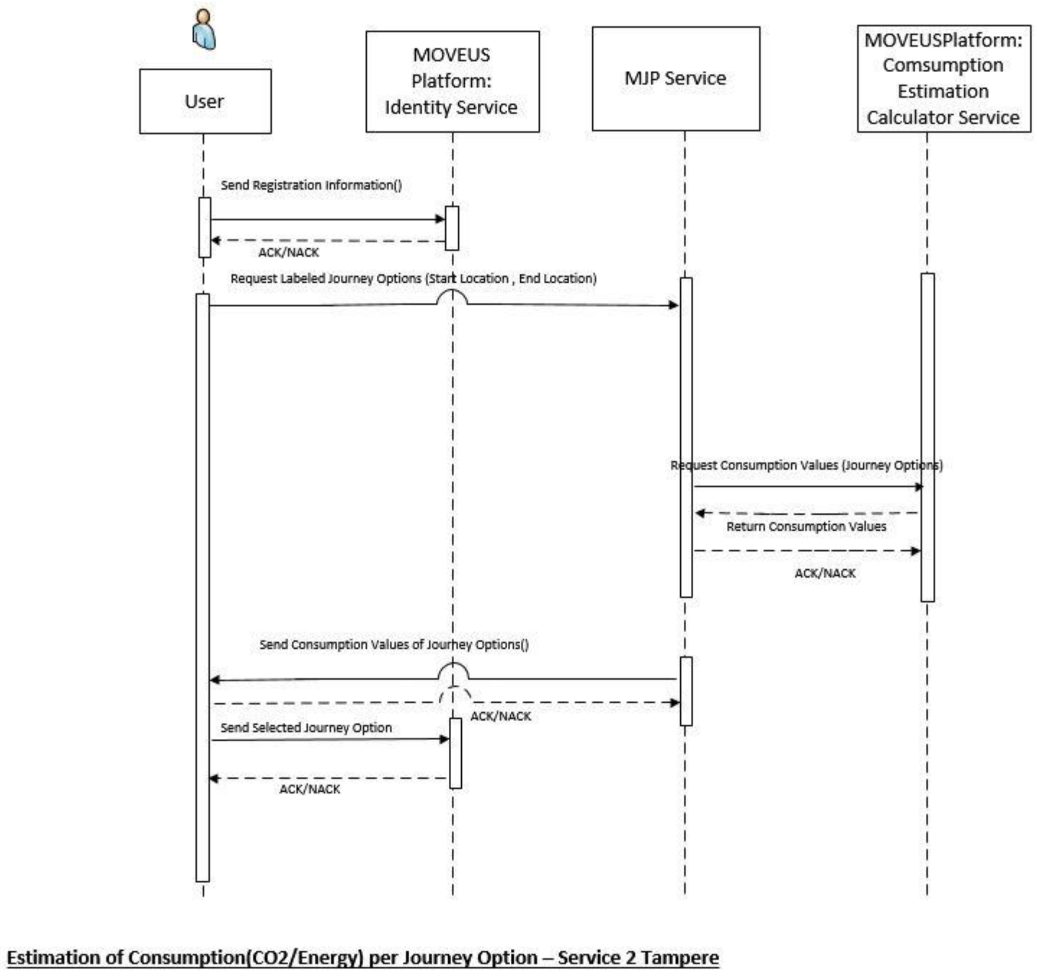

User
MOVEUS Platform: Identity Service
MJP Service
MOVEUSPlatform: Comsumption Estimation Calculator Service
Send Registration Information()
ACK/NACK
Request Labeled Journey Options (Start Location , End Location)
Request Consumption Values (Journey Options)
Return Consumption Values
ACK/NACK
Send Consumption Values of Journey Options()
ACK/NACK
Send Selected Journey Option
ACK/NACK
Estimation of Consumption(CO2/Energy) per Journey Option – Service 2 Tampere

Figure 31. *Sequence diagram of Estimation of Consumption (CO_2/Energy) per Journey Options.*

The following steps describes the flow of request and response of the sequence diagram in the Figure 31:

1. ***User → MoveUs platform***
   - Enter *MoveUs* web site
   - Enter username and password and start the registration; a preliminary email for confirmation will be sent to user to check email address.
   - Enter information on user *(→ 2) :*
     – Name, address, VAT
     – Special roles (e.g. tourist, business)
     – Direct influencers of personal mobility choices (e.g. passion for biking / running, personal acceptance of car/bike sharing)
     – Indirect influencers of personal mobility choices (e.g. preference to commute to work / school in groups)
     – Direct influencers of personal routing choices (e.g. aversion towards specific places in the city)

  - Habits (e.g. regular itineraries undertaken)
  - Particularities of personal vehicles the user is regularly driving (fuel needs, consumption, etc.)

2. ***User → Multi-modal Journey Planner service (Tampere)***
   - Enter information on START LOCATION and END LOCATION
   - Request Journey Option(s) and consumption labels associated to provided Start and End coordinates (i.e. the Journey Options consumption information) *(→ 3)*
3. ***Multi-modal Journey Planner service (Tampere)***
   - Calculate Journey Option(s) associated to provide input Start and End coordinates *(→ 4)*
4. ***Multi-modal Journey Planner service (Tampere)→ Consumption Estimation Calculator service***
   - Provide Journey Option(s) to label with consumption values
   - Request Journey Option(s) consumption information *(→ 5)*
5. ***Consumption Estimation Calculator service***
   - Computes an estimation of consumption of energy/CO_2 associated to input Journey Option(s) *(→ 6)*
6. ***Consumption Estimation Calculator service →Multi-modal Journey Planner service (Tampere)***
   - Provide Journey Option(s) consumption information *(→ 7)*
7. ***Multi-modal Journey Planner service (Tampere) → User***
   - Provide Journey Option(s) consumption information *(→ 8)*
8. ***User***
   - Selects one Journey Option. *(→ 9)*
9. ***MoveUs Platform: Identity Service***
   - Stores information about user's selected Journey Option(s) and associated consumptions (for development of long-term statistics/consumption stores information about user's selected

3.4.2.3 User tailored incentive-based visualization of Journey Options

This service aims to give an incentive oriented view of input transportation options information, per user. It acts as an adaptor for user-friendly meaningful display of backend computed information.

In *MoveUs* for the Tampere pilot, this service targets the mapping of EE/CO_2 labels to incentive points (and subsequently relevant incentives) for the user considered, based on a Set of Incentive Rules stored in a dedicated DB. This service finally displays an incentive-based set of computed EE/CO_2 journey options. In this thesis work, the service is embedded into the architecture and it is depicted via sequence diagram shown in the Figure 32.

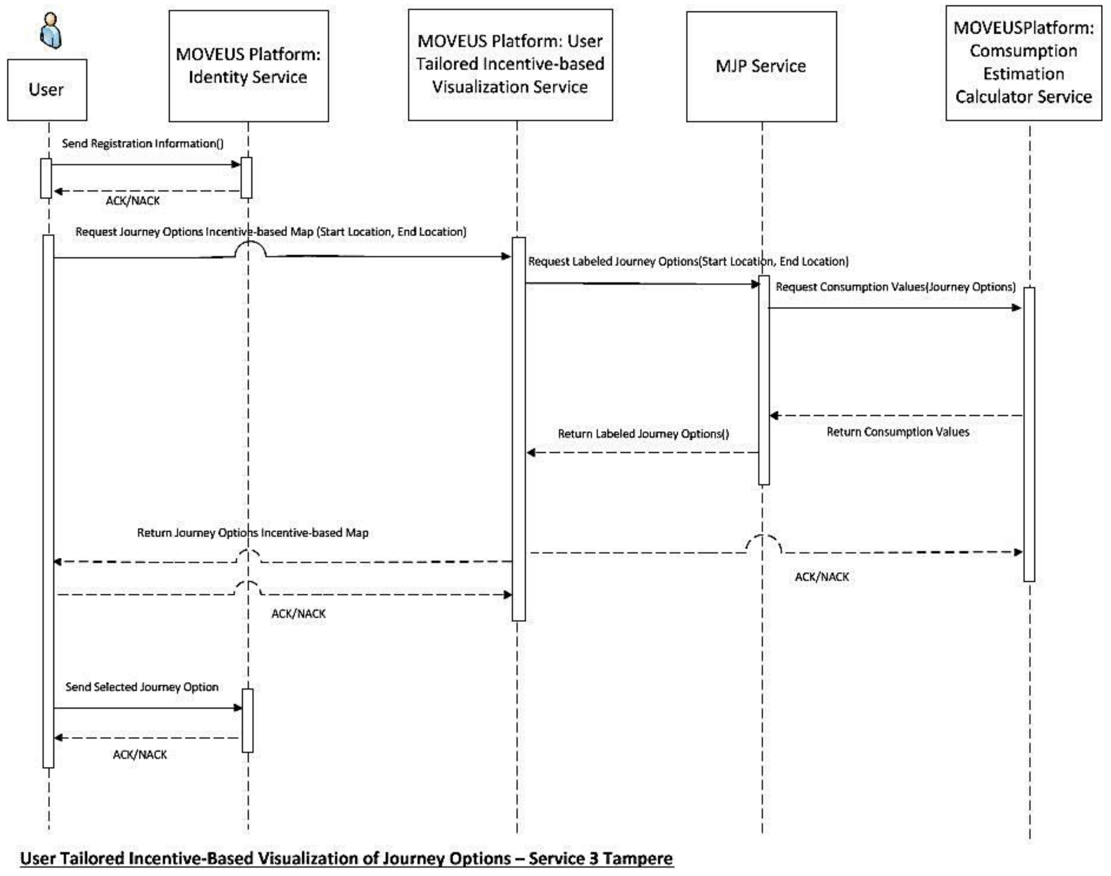

User
MOVEUS Platform: Identity Service
MOVEUS Platform: User Tailored Incentive-based Visualization Service
MJP Service
MOVEUSPlatform: Comsumption Estimation Calculator Service
Send Registration Information()
ACK/NACK
Request Journey Options Incentive-based Map (Start Location, End Location)
Request Labeled Journey Options(Start Location, End Location)
Request Consumption Values(Journey Options)
Return Consumption Values
Return Labeled Journey Options()
Return Journey Options Incentive-based Map
ACK/NACK
ACK/NACK
Send Selected Journey Option
ACK/NACK
User Tailored Incentive-Based Visualization of Journey Options – Service 3 Tampere

Figure 32. *Sequence diagram of User Tailored Incentive-Based Visualization of Journey Options.*

The following steps describes the flow of request and response of the sequence diagram in the Figure 32:

1. ***User → MoveUs platform***
   - Enter *MoveUs* web site
   - Enter username and password and start the registration; a preliminary email for confirmation will be sent to user to check email address.
   - Enter information on user (→ 2):
     - Name, address, VAT
     - Special roles (e.g. tourist, business)
     - Direct influencers of personal mobility choices (e.g. passion for biking / running, personal acceptance of car/bike sharing)
     - Indirect influencers of personal mobility choices (e.g. preference to commute to work / school in groups)
     - Direct influencers of personal routing choices (e.g. aversion towards specific places in the city)
     - Habits (e.g. regular itineraries undertaken)
     - Particularities of personal vehicles the user is regularly driving (fuel needs, consumption, etc.)

2. ***User → MoveUs platform: User-tailored incentive-based visualization service***
   - Enter information on START LOCATION and END LOCATION
   - Request Journey Options Incentive-based Map (i.e. a visual-friendly display of consumption labels associated to provided Start and End coordinates) (→3)
3. ***MoveUs Platform: User-tailored incentive-based visualization service →Multi-modal Journey Planner service (Tampere)***
   - Request Journey Option(s) Consumption Information associated to given input Start and End coordinates (→4)
4. ***Multi-modal Journey Planner service (Tampere) → Consumption Estimation Calculator service***
   - Provide Journey Option(s) to be evaluated from the viewpoint of consumption (CO_2/Energy)
   - Request Journey Option(s) Consumption Information (→5)
5. ***MoveUs Platform: Consumption Estimation Calculator service***
   - Computes an estimation of consumption of energy/CO_2 associated to input Journey Option(s) (→6)
6. ***MoveUs Platform: Consumption Estimation Calculator service → MoveUs Platform: User tailored incentive-based visualization service***
   - Provides quantification of the estimated amount of energy/CO_2 associated to one/more Journey Option(s) (→7)
7. ***MoveUs Platform: User tailored incentive-based visualization service***
   - Identifies the optimal, most efficient journey option, from the viewpoint of CO_2/Energy consumption
   - Identifies the difference (in energy/CO2 consumption values) between each of the other journey options and the optimal one, and translates this difference to a number / color code (→8)
8. ***MoveUs Platform: User tailored incentive-based visualization service → User***
   - Provides color codes to visualize the journey options on a map, by taking into account the user profile and the incentives available
   - Provide Journey Options Incentive-based Map (i.e. information on consumption associated to Journey Option(s) between Start and End coordinates/location and color codes associated to incentives)(→ 9)
9. ***User***
   - Selects one of the available Journey Option(s) (→ 10)
10. ***MoveUs Platform: Identity Service***
    - Stores information about user's selected Journey Option(s) and the incentives associated to the selection.

4. IMPLEMENTATION

This chapter shows details of the design and implementation of UML diagrams for the proposed architecture in this thesis work. Section 4.1 provides details about the requirements for the problem to be solved in Tamper pilot section. Details about the design of Data Access Objects and Persistence layer is covered in section 4.2. This section also demonstrates how J2EE, hibernate framework and Junit framework are utilized to implement the system in this thesis work.

4.1 Requirements specification for the problem to be solved in Tampere

The below Figure 33 illustrates the use case designed for Tampere pilot. The Consumption Estimation Calculator comprise of energy labels (static), energy Key Performance Indicator (KPI) and computational methods and estimation algorithm engine. The consumption estimator provides optimum journey options based on the EE/CO_2 assessments with respect to the input data provided by Multi-modal Journey Planner (MJP). The MJP comprises of different journey planners such as bus, cycling and car. These journey planners consume the API from various databases such as SIRI & DATEX-II. Table 6 shows types of innovative services created in this thesis work to solve the problem.

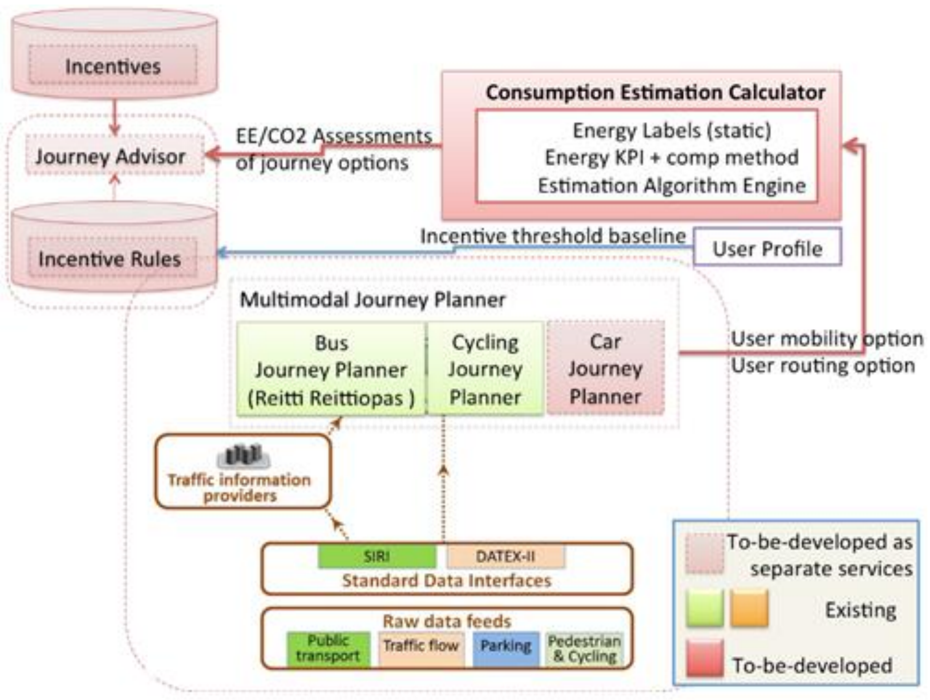


Figure 33. *Overall structure of MoveUs services to be developed in Tampere pilot.*

Based on the journey options provided by the Consumption Estimation Calculator, Incentives and Incentive Rules the journey advisor provides optimum user specific journey options.

Table 6. *Services in Tampere pilot.*

Use cases in Tampere pilot
1. Calculation of Multi-modal Journey Options
2. Estimation of Consumption (CO_2 and / or Energy) per Journey Option
3. User tailored incentive-based visualization of Journey Options

4.2 Implementation of DAO and Persistence Layer

Data Access Object (DAO) is an object that provides an abstract interface to some type of database or other persistence mechanism. DAO is responsible for data access from the persistence layer and manipulation of data in the persistence layer [48].

4.2.1 Class Diagrams of DAO developed

4.2.1.1 Vehicle Class Diagram

Vehicle class diagram is a representative of all types of vehicles in the system. This class diagram explains the *Vehicle* base-class and sub-class of diverse types of vehicle such as *Bike*, *Car* and *EmergencyVehicle*. It also elaborates the association between the *FuelNeed* and *Vehicle* base-class.

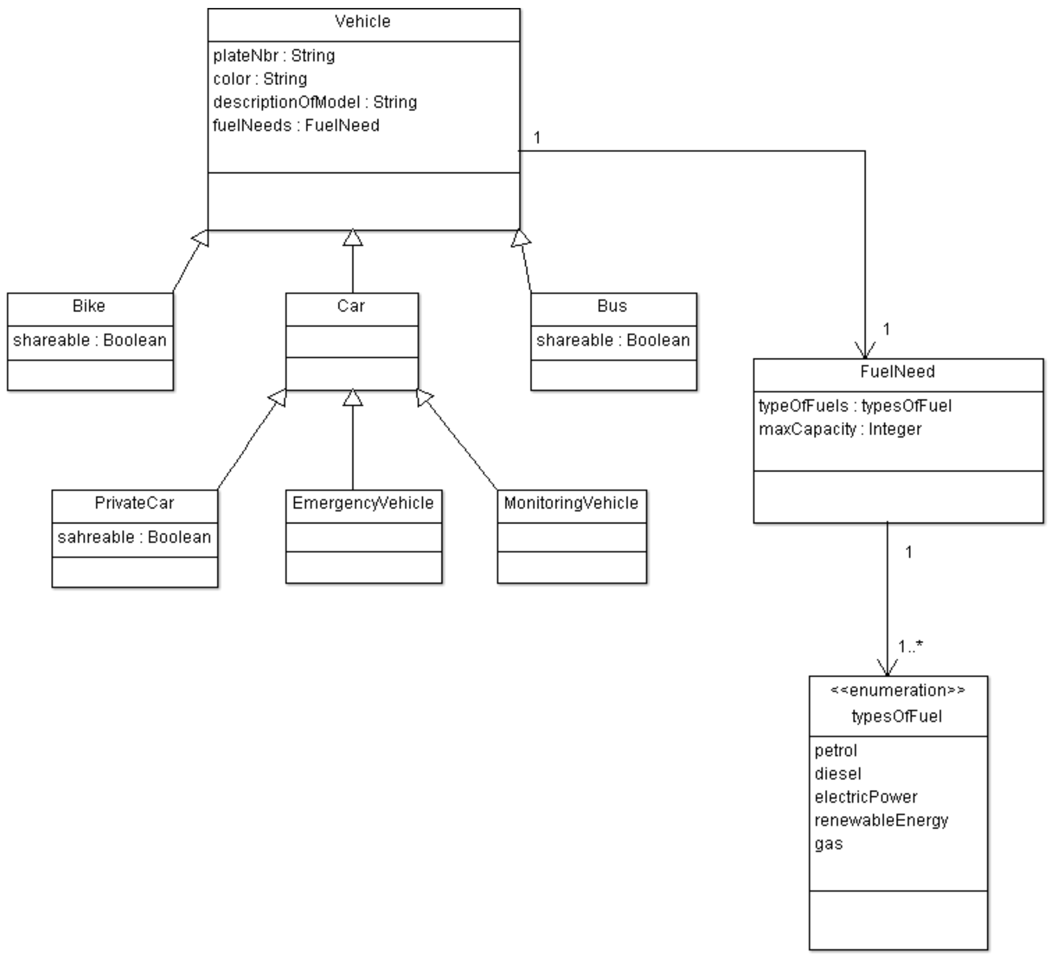


Figure 34. *Vehicle Class Diagram.*

4.2.1.2 VehicleUser Class Diagram

VehicleUser class diagram is a representative of different kinds of vehicle users in the system. This class diagram shows the *VehicleUser* base-class and sub-class of different types of vehicle users such as *BikeUser* and *CarUser*. These sub-classes are also associated with respective vehicle classes in one to many relationship fashion. It also elaborates the association between the *InfoPiece* and *Vehicle* base-class.

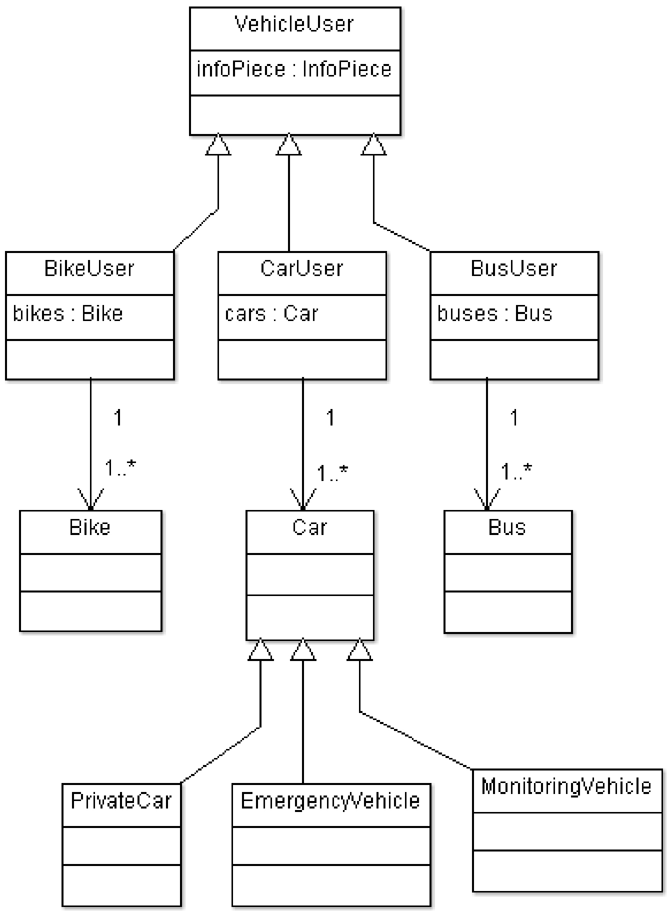


Figure 35. *VehicleUser Class Diagram.*

4.2.1.3 InfoPiece Class Diagram

It is a class view of all information’s combined from various data source to enhance the usability of *MoveUs* application. It basically comprises of *InfoPiece* base-class and sub-class of various other *InfoPiece’s* such as *MapInfoPiece*, *NavigationOptionInfo* and *TrafficFlow*. This conceptual class view is created based on the available data-sets from various sources.

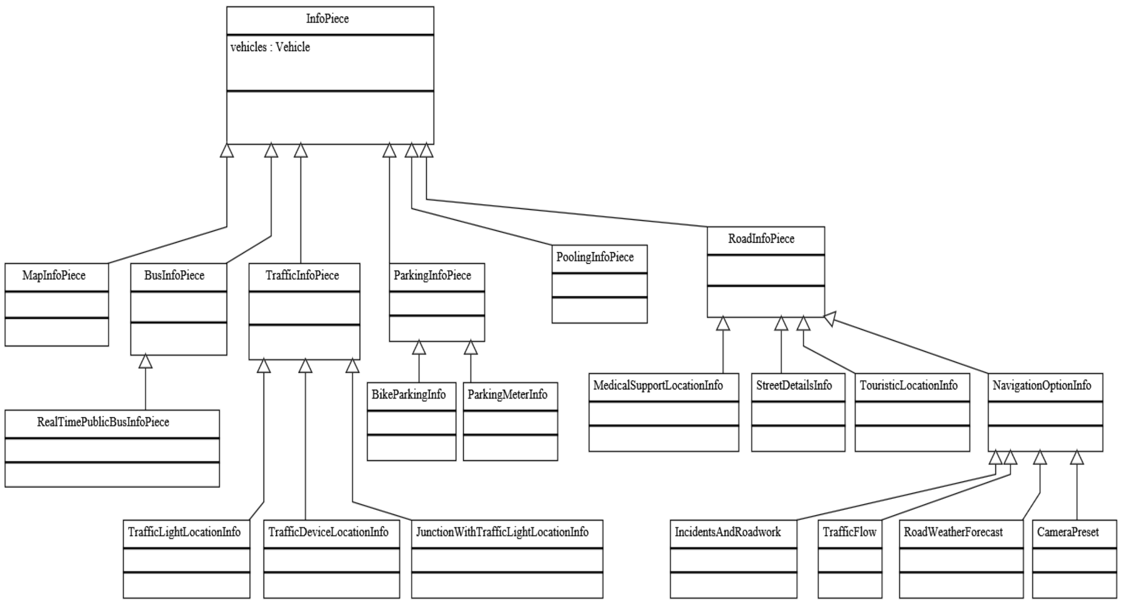


Figure 36. *InfoPiece Class Diagram.*

4.2.1.4 KeyPerformanceIndicator Class Diagram

A Key Performance Indicator (KPI) is a business metric used in a system to evaluate factors that are crucial in achieving the key business objectives of that system. The business objective of this system is to provide incentives based on the carbon footprint, routing options and mobility options. The *KeyPerformanceIndicator* base-class is associated with *InfoPiece* in order to access the required information from relevant *InfoPiece* which determine the KPI values.

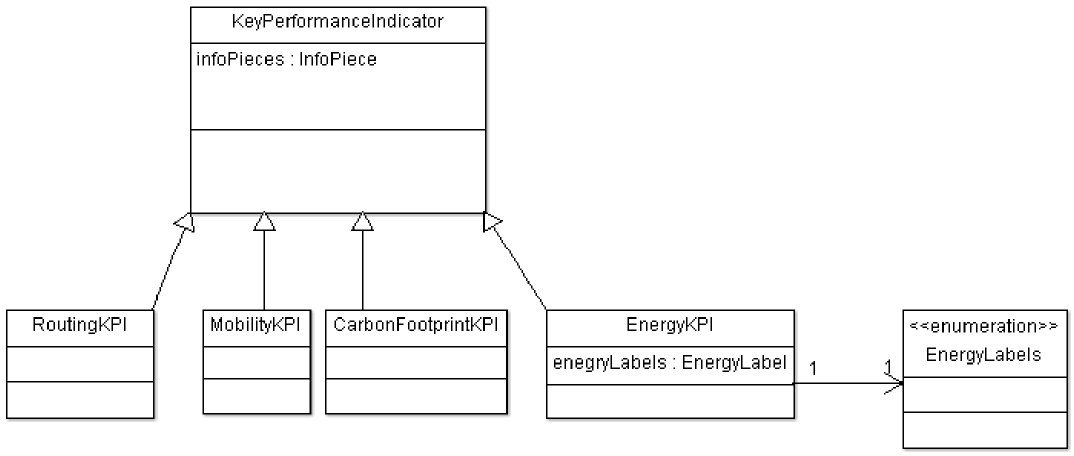


Figure 37. *KeyPerformanceIndicator Class Diagram.*

4.2.1.5 NavigationRecommendation Class Diagram

NavigationRecommendation class diagram illustrates how the navigation recommendations for each *MoveUs* user's is designed at object design level. Each *MoveUs* user can avail may navigation recommendations based on values created by *ConsumptionEstimationCalculator* class.

The profile of each user is stored in the system which contains personal information's such as name, different address and phone numbers.

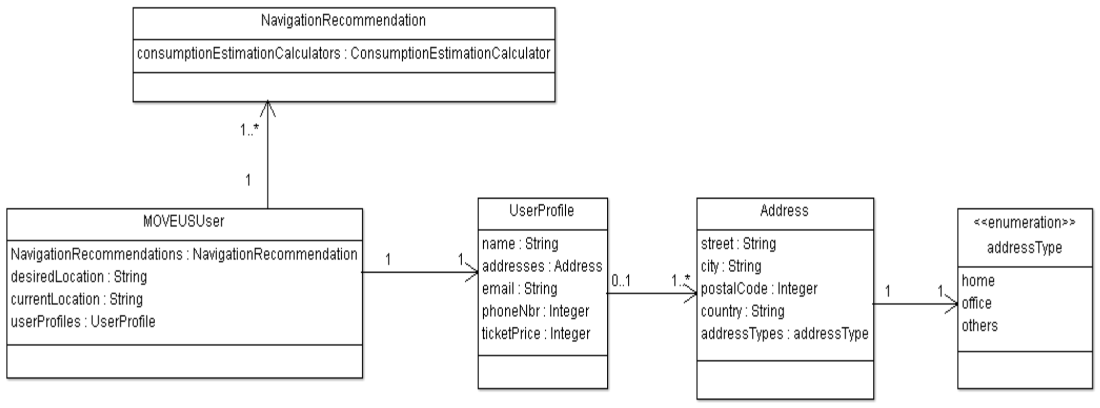


Figure 38. *NavigationRecommendation Class Diagram.*

4.2.1.6 ConsumptionEstimationCalculator Class Diagram

The figure 34 demonstrates the class diagram of *ConsumptionEstimationCalculator* and the relationship between *KeyPerformanceIndicator*, *InfoPiece*, *VehicleUser*, *Vehicle* and *NavigationRecommendation*. The *ConsumptionEstimationCalculator* class view shows based on what information's the KPI's values are generated. One of the main attribute of *ConsumptionEstimationCalculator* is *keyPerformanceIndicators*, based on its value multiple journey options are created according to each user's perspective.

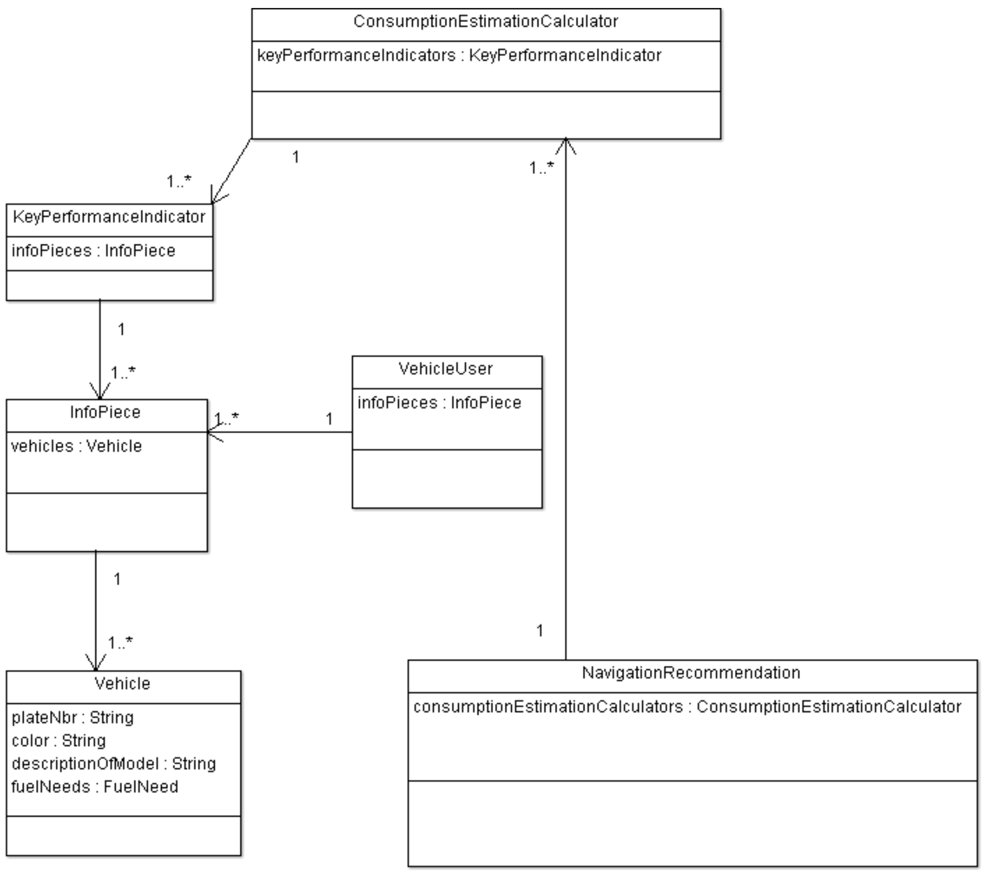


Figure 39. *ConsumptionEstimationCalculator Class Diagram.*

4.2.2 Implementation of Data Persistence Layer Based on Hibernate

In this thesis work, hibernate was used to implement the data persistence layer over the DAO of mobility system. The system development environment is selected as follows: eclipse IDE, JDK kit, Tomcat 7.0 and Derby database. The system data is persisted by Hibernate, which can be created by following steps.

4.2.2.1 Creating Persistence Class

In order to realize the relational data mapping, the first step is to create the persistent class. They are usually the domain model entity classes. And every instances of entity classes will be stored in database tables. Hibernate functions best if these persistent classes follow the Plain Old Java Object (POJO). The codes of *VehicleUser* persistence class are as follows.

```java
import java.util.HashSet;
import java.util.Set;

public class VehicleUser {

    private long Id;
    private Set<InfoPiece> infoPieces = new HashSet<InfoPiece>();

    public Set<InfoPiece> getInfoPieces() {
        return infoPieces;
    }

    public void setInfoPieces(Set<InfoPiece> infoPieces) {
        this.infoPieces = infoPieces;
    }

    public void setId(long id) {
        Id = id;
    }

    public long getId() {
        return Id;
    }
}
```

4.2.2.2 Creating ORM Mapping

In Hibernate framework, Object/Relational Mapping mechanism are usually defined in an XML formatted file. In this thesis work Hibernate annotations are used to define mappings instead of using XML formatted file. Hibernate Annotation used to club all the metadata into the Plain Old Java Object (POJO) java file along with the code this creates the table structure and POJO simultaneously during the development. The following code is the mapping of *VehicleUser* class with annotations to map objects with the defined *VEHICLEUSER* table:

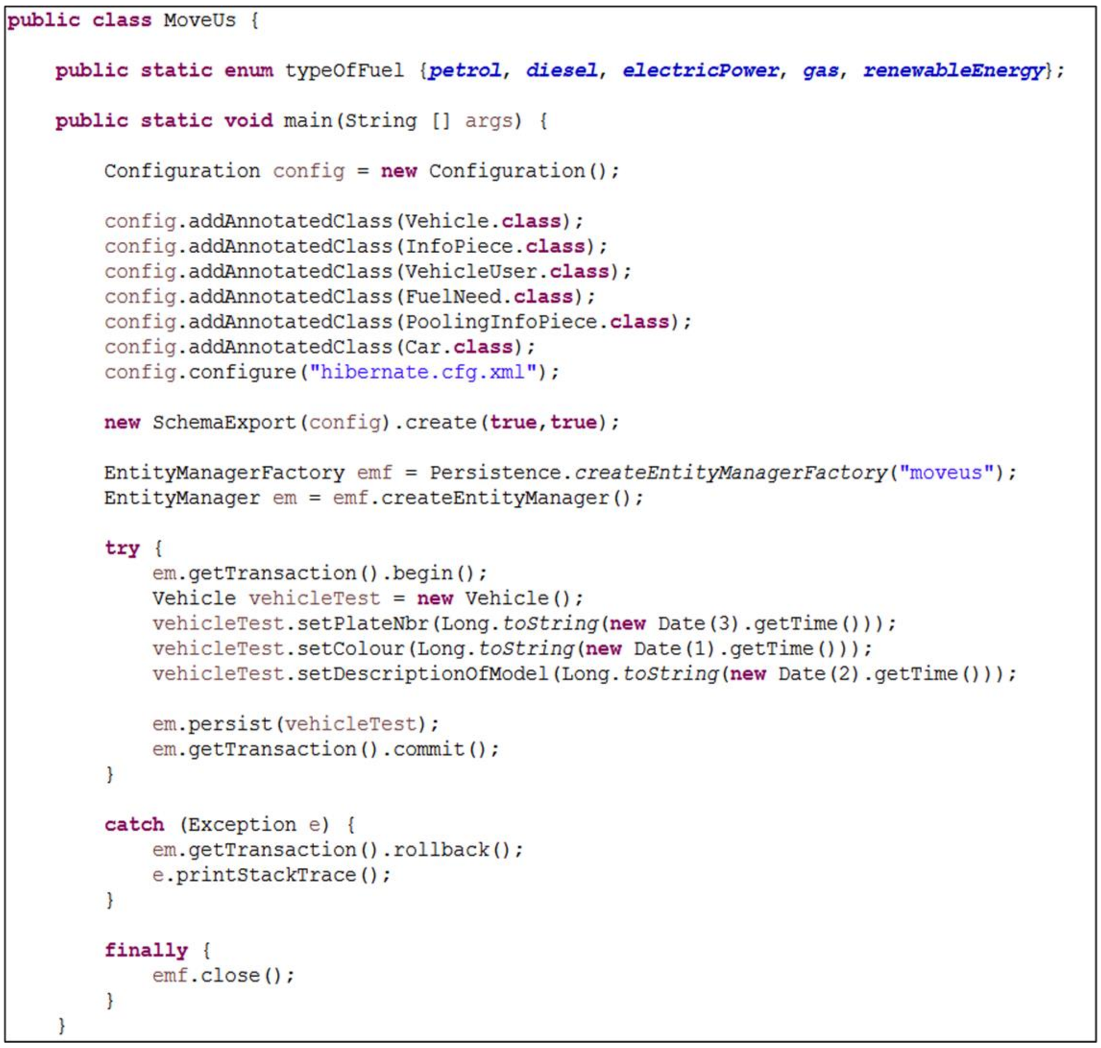
```java
import java.util.HashSet;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "VEHICLEUSER")
public class VehicleUser {

    @Id
    private long Id;

    @OneToMany(mappedBy = "vehicleUser")
    private Set<InfoPiece> infoPieces = new HashSet<InfoPiece>();

    public Set<InfoPiece> getInfoPieces() {
        return infoPieces;
    }

    public void setInfoPieces(Set<InfoPiece> infoPieces) {
        this.infoPieces = infoPieces;
    }

    public void setId(long id) {
        Id = id;
    }

    public long getId() {
        return Id;
    }
}
```

The hibernate.cfg.xml configuration file is created to define database related information. The following code snippet shows the database configuration:

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory name="java:hibernate/SessionFactory">

        <property name="hibernate.connection.url">jdbc:derby://localhost:1527/HibernateDb;create=true</property>
        <property name="hibernate.connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <property name="hibernate.dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>
        <property name="hibernate.connection.username">rajeshr</property>
        <property name="hibernate.connection.password">rajeshr123</property>
        <property name="hibernate.default_schema">MoveUs</property>
        <property name="hibernate.transaction.factory_class">org.hibernate.transaction.JDBCTransactionFactory</property>
        <property name="hibernate.current_session_context_class">thread</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.hbm2ddl.auto">update</property>

    </session-factory>
</hibernate-configuration>
```

4.2.2.3 Creating Application Class

Finally, the application class is created with the *main ()* method to run the application.



```java
public class MoveUs {

    public static enum typeOfFuel {petrol, diesel, electricPower, gas, renewableEnergy};

    public static void main(String [] args) {

        Configuration config = new Configuration();

        config.addAnnotatedClass(Vehicle.class);
        config.addAnnotatedClass(InfoPiece.class);
        config.addAnnotatedClass(VehicleUser.class);
        config.addAnnotatedClass(FuelNeed.class);
        config.addAnnotatedClass(PoolingInfoPiece.class);
        config.addAnnotatedClass(Car.class);
        config.configure("hibernate.cfg.xml");

        new SchemaExport(config).create(true,true);

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("moveus");
        EntityManager em = emf.createEntityManager();

        try {
            em.getTransaction().begin();
            Vehicle vehicleTest = new Vehicle();
            vehicleTest.setPlateNbr(Long.toString(new Date(3).getTime()));
            vehicleTest.setColour(Long.toString(new Date(1).getTime()));
            vehicleTest.setDescriptionOfModel(Long.toString(new Date(2).getTime()));

            em.persist(vehicleTest);
            em.getTransaction().commit();
        }

        catch (Exception e) {
            em.getTransaction().rollback();
            e.printStackTrace();
        }

        finally {
            emf.close();
        }
    }
```

4.2.3 The Unit Testing of Data Persistence Layer Based on Hibernate

After creating the persistence layer, the unit testing becomes one of the indispensable step. It mainly used to detect whether the interface methods are correctly correlated. The interface is also checked by modifying the relevant method when the new method is added or the old method is changed.

4.2.3.1 Creating Base Testing Class

First of all, the class that named *VehicleTest* is created to test *Vehicle* class. In order to use *JUnit* annotations, *JUnit* plugins should be installed along with *eclipse* IDE. Once *JUnit* is installed the *@BeforeClass* annotation supported by *JUnit* can be used. This annotation indicates that the static method to which it is attached must be executed once and before all tests in the class. And annotating a public static void method with *@AfterClass* causes that method to be run after all the tests in the class have been run. All *@AfterClass* methods are guaranteed to run even if a method which is annotated with *@BeforeClass* throws an exception. And the *@AfterClass* methods declared in super classes will be run after those of the current class. The methods which are used to test the class are annotated with *@Test*. *VehicletTest* method is used to test the attributes and *getter* and *setter* methods of those attributes pertaining to *Vehicle* class. The bellow code snippet shows the *VehicleTest* class with three annotated static classes which are executed before and after testing *testVehicle*:

```
public class VehicleTest {
    private static EntityManager entityManager;

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        EntityManagerFactory factory = Persistence
                .createEntityManagerFactory("moveus");
        entityManager = factory.createEntityManager();
        entityManager.getTransaction().begin();
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        entityManager.close();
        System.out.println("working");
    }
    @Test
    public void testVehicle() {
        try {
        Vehicle vehicleTest = new Vehicle();
        System.out.println("IS ACTIVE");
        vehicleTest.setPlateNbr(Long.toString(new Date(0).getTime()));
        vehicleTest.setColour(Long.toString(new Date(1).getTime()));
        vehicleTest.setDescriptionOfModel(Long.toString
                (new Date(2).getTime()));

        entityManager.persist(vehicleTest);

        System.out.println("vehicle" + vehicleTest + ","
                + " vehicle.Id=" + vehicleTest.getId());
        Vehicle foundVehicle = entityManager.find
                (Vehicle.class, vehicleTest.getId());
        System.out.println("foundVehicle = " + foundVehicle);

        assertEquals(vehicleTest.getPlateNbr(),
                foundVehicle.getPlateNbr());
        assertEquals(vehicleTest.getColour(),foundVehicle.getColour());
        assertEquals(vehicleTest.getDescriptionOfModel(),
                foundVehicle.getDescriptionOfModel());
        }

        catch (Exception e) {
            entityManager.getTransaction().commit();
            entityManager.getTransaction().rollback();
            e.printStackTrace();
        }
    }
```

4.2.3.2 Testing Interrelated Method

Once the test class is created with initial methods, the main methods which are created to test the interface with a particular class is annotated with *@Test* annotation. The method *testLinkVehicleUserInfopiece* class is used to test the *ManyToOne* mapping relation between *Vehicle* and *Infopiece* object. The following code piece contains the *@Test* annotated method *testVehicleAndInfopiece:*

```
@Test
public void testVehicleAndInfoPiece() {
    try {

        Vehicle vehicleTest = new Vehicle();
        InfoPiece infoPieceTest = new InfoPiece();
        entityManager.getTransaction().commit();

        entityManager.persist(vehicleTest);
        entityManager.persist(infoPieceTest);

        assertEquals(null,vehicleTest.getInfoPiece());
        vehicleTest.setInfoPiece(infoPieceTest);

        entityManager.merge(vehicleTest);
        assertNotNull(vehicleTest.getInfoPiece());
        System.out.println("IS ACTIVE");
    }

    catch (Exception e) {

        entityManager.getTransaction().rollback();
        e.printStackTrace();
    }
}
```

Every test method created in this work contains test case for fail situations as well. The bellow code snippet shows one of the fail case testing method *testVehicleFailCase* from *VehicleTest* class:

```
@Test
public void testVehicleFailCase() {

    try {

        Vehicle vehicleTest = new Vehicle();

        System.out.println("IS ACTIVE");
        vehicleTest.setPlateNbr(Long.toString(new Date(0).getTime()));
        vehicleTest.setColour(Long.toString(new Date(1).getTime()));
        vehicleTest.setDescriptionOfModel(Long.toString(new Date(2).getTime()));

        entityManager.persist(vehicleTest);

        System.out.println("vehicle" + vehicleTest + ", vehicle.Id=" +
                            vehicleTest.getId());
        Vehicle notFoundVehicle = entityManager.find(Vehicle.class,
                                    vehicleTest.getId());
        System.out.println("Plate Number = " +
                            notFoundVehicle.getPlateNbr());
        System.out.println("Color = " +
                            notFoundVehicle.getColour());
        System.out.println("Description Of Model = " +
                            notFoundVehicle.getDescriptionOfModel());
        assertEquals(1, notFoundVehicle.getPlateNbr());
        assertEquals(2, notFoundVehicle.getColour());
        assertEquals(3, notFoundVehicle.getDescriptionOfModel());

        fail("Vehicle's Plate Number Not Found");
    }

    catch (Exception e) {
        entityManager.getTransaction().commit();
        entityManager.getTransaction().rollback();
        e.printStackTrace();
    }
}
```

5. CONCLUSIONS

In this chapter, conclusions are derived grounded on the results of implemented DAO and persistence layer. Section 5.1 contains contribution of this thesis work and discussion about the implemented architecture. And section 5.2 contains recommendation for future work in this domain.

5.1 Contribution and Discussion

For this research work, three main services are modeled which are calculation of *Multimodal Journey Options*, *Estimation of Consumption per Journey Options* object, and *User tailored incentive-based visualization of Journey Options.* Data access object and persistence layer have been created for the proposed services in this thesis. The implemented service models reduce the carbon footprint by allowing the user to choose incentive based mobility options and routing options.

The objective of this thesis is to model the use cases and implement the data persistence for a multi-modal intelligent transportation architecture which improves methods of generating energy efficient journey options. The developed use cases evidently fulfill these requirements. It also contains an incentive-based visualization service which provides incentives for the users who opts for mobility and routing options which consumes less EE/CO_2. The main business objective of this service is to provide incentives defined by key performance indicators based on the *carbon footprint*, *routing options* and *mobility options*. This motivates the user to opt mobility and routing options with less energy consumption, suggested by the *MoveUs* application. The demonstrated services incorporate dynamic traffic information from local traffic management system for calculating the mobility and routing options with minimum energy consumption. And also other useful information such as traffic light details, road weather forecast, location of public bus at real time and parking information.

In this work, the architecture is modeled using object oriented process and UML approach. The services are described using use case and sequence diagrams. Data access object and persistence layer is designed for the proposed architecture in Java and hibernate framework, and finally the persistence is tested using unit test framework, Junit. In this work Hibernate annotations are used to define mappings instead of using XML formatted file. The testing is performed to detect whether the interface methods are correctly correlated. The interface is also checked by modifying the relevant method when the new method is added or the old method is changed.

5.2 Future work

In future, incorporating behavioral differences between users while planning a multi-modal will be of great significance. Because for some users due to some personal reasons it is not possible to take a public bus while for others this might be the best option. The possibility to avail such functionality is based on a sufficient data which has to be collected in real-world scenarios. Also, it is possible to give *cross-prioritization* to the pedestrian users. This type of *smart crossing* can avoid risk of accidents and also reduce travel time for vehicle users who wait for green traffic signal, even if there are no pedestrians to cross. This service is can be implemented if there are changes at infrastructure level of transportation.

REFERENCES

[1] U.S Energy Information Administration [WWW]. Available on: http://www.eia.gov/

[2] Commission of the European Communities, “Green Paper on urban mobility: Towards a new culture for urban mobility” COM 551, 2007 [WWW]. Available on: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52007DC0551

[3] Proposal for a regulation of the European Parliament and of the Council amending Regulation (EU) No 443/2009 to “Define the modalities for reaching the 2020 target to reduce CO_2 emissions from new light commercial vehicles” [WWW]. Available on: http://eur-lex.europa.eu/resource.html?uri=cellar:70f46993-3c49-4b61-ba2f-77319c424cbd.0001.02/DOC_1&format=PDF

[4] International Energy Agency, Finland 2013 Review [WWW].Available on: https://www.iea.org/publications/freepublications/publication/Finland2013_free.pdf.

[5] J. M. Sussman, *Perspectives on Intelligent Transportation Systems (ITS)*. New York: Springer-Verlag, 2005.

[6] M. Barth and M. Todd, "Intelligent transportation system architecture for a multi-station shared vehicle system," *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, Dearborn, MI, 2000, pp. 240-245.

[7] M. Barth, Jing Han and M. Todd, "Performance evaluation of a multi-station shared vehicle system," *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, Oakland, CA, 2001, pp. 1218-1223.

[8] J. Shawe-Taylor, T. De Bie and N. Cristianini, "Data mining, data fusion and information management," in *IEEE Proceedings - Intelligent Transport Systems*, vol. 153, no. 3, pp. 221-229, September 2006.

[9] J. D. Vreeswijk, M. K. M. Mahmod and B. van Arem, "Energy efficient traffic management and control - the eCoMove approach and expected benefits," *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, Funchal, 2010, pp. 955-961.

[10] Repowering Transport Project White Paper April 2011 [WWW]. Available on: http://www3.weforum.org/docs/WEF_RepoweringTransport_ProjectWhitePaper_2011.pdf

[11] Ferreira, J. T., Paulo, Porfírio Filipe, “Collaborative Car Pooling,” *International Conference on Sustainable Urban Transport and Environment*, Paris, 2009.

[12] European Commission Extra Consortium for DG Energy and Transport, 2001.

[13] J. Zhang, F. Y. Wang, K. Wang, W. H. Lin, X. Xu and C. Chen, "Data-Driven Intelligent Transportation Systems: A Survey," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624-1639, Dec. 2011.

[14] T. Kadoono, T. Yamaka, R. Shibasaki and K. Amma, "An analysis on multimodal transportation of international maritime container cargo in Japanese hinterland," *OCEANS '04. MTTS/IEEE TECHNO-OCEAN '04*, Kobe, 2004, pp. 2270-2275 Vol.4.

[15] U.S. Department of Transportation, “The National ITS Architecture Version 5.1”, 2005.

[16] Ministry of Internal Affairs and Communication National Police Agency, and Ministry of Land, Infrastructure, and Transport of Japan, “Vehicle Information and Comm. System,” 2006 [WWW]. Available On: http://www.vics.or.jp/englishindex.html.

[17] European Commission, "The KAREN European ITS Framework Architecture," 2004 [WWW]. Available On: http://frame-online.eu/.

[18] Research and Innovation Technology Administration (RITA), US Department of Transportation, “National ITS Architecture Theory of Operations,” pp 154. [WWW]. Available on: http://www.iteris.com/itsarch/documents/theory/theory.pdf.

[19] A. Arrayangkool and A. Unakul, "A flexible Intelligent Transportation System architecture model with object oriented methodology and UML," *Communications and Information Technology, 2009. ISCIT 2009. 9th International Symposium on*, Icheon, 2009, pp. 741-746.

[20] European ITS framework architecture, “FRAME Architecture Version 4.1” [WWW]. Available On: http://frame-online.eu/wp-content/uploads/2015/09/D15-FRAME-Architecture-Part-1-1.0.pdf, pp. 17.

[21] Yonghua Zhou, Yuliu Chen and Huapu Lu, "UML-based systems integration modeling technique for the design and development of intelligent transportation management system," *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 2004, pp. 6061-6066 vol.7.

[22] M. Prandtstetter, M. Straub and J. Puchinger, "On the way to a multi-modal energy-efficient route," *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, Vienna, 2013, pp. 4779-4784.

[23] E. W. Dijkstra, “A note on two problems in connexion with graphs,” Numerische Mathematik, vol. 1, pp. 269–271, 1959.

[24] Eva Ericsson, Hanna Larsson, Karin Brundell-Freij, “Optimizing route choice for lowest fuel consumption – Potential effects of a new driver support tool,” *Transportation Research*, Part C: Emerging Technologies, Volume 14, Issue 6, December 2006, Pages 369-383, ISSN 0968-090X.

[25] K. Kraschl-Hirschmann and M. Fellendorf, “Estimating energy consumption for routing algorithms,” in Intelligent Vehicles Symposium. IEEE, 2012, pp. 258–263.

[26] D. Delling, J. Dibbelt, T. Pajor, D. Wagner, and R. F. Werneck, “Computing multi-modal journeys in practice,” in Experimental Algorithms, ser. LNCS, V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela, Eds. Springer Berlin Heidelberg, 2013, vol. 7933, pp. 260–271.

[27] J. Zhang, F. Liao, T. Arentze, and H. Timmermansa, “A multi-modal transport network model for advanced traveler information systems,” *Procedia Computer Science*, vol. 5, no. 0, pp. 912 – 919, 2011, the 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011).

[28] H. Yu and F. Lu, “A Multi-Modal Route Planning Approach with an Improved Genetic Algorithm,” in *Joint International Conference on Theory, Data Handling and Modeling in GeoSpatial Information Science*, 2010, pp. 343–348.

[29] J. C. Ferreira, P. Filipe and A. Silva, "Multi-Modal Transportation Advisor system," *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*, Vienna, 2011, pp. 388-393.

[30] D. K. W. Chiu, O. K. F. Lee, Ho-fung Leung, E. W. K. Au and M. C. W. Wong, "A Multi-Modal Agent Based Mobile Route Advisory System for Public Transport Network," *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 2005, pp. 92b-92b.

[31] Finland’s Strategy for Intelligent Transport 2009 [WWW]. Available On: http://www.lvm.fi/documents/20181/817459/Programmes+and+strategies+6-2009/c32cbc43-fb94-45c5-a9d9-208fa4cbc37f?version=1.0.

[32] R P Sriganesh, G. Brose, M. Silverman, "Mastering Enterprise JavaBeans 3. 0," John Wiley & Sons, Inc. New York, NY, USA, 2006,pp. 21- 25.

[33] Roland Barcia, "Get to know Java EE 5," developerWorks, Aug 2007.

[34] S. Ambler, Agile Database Techniques: Effective Strategies for the Agile Software Developer, USA: Wiley, 2003.

[35] Li Gang, The application of Lightweight Java EE framework[M]. Beijing: Electronic Industry Press,2008, pp. 541-560.

[36] HaiLan Pan, AnBao Wang and WenRong Jiang, "Discussion of course of E-commerce website construction based on java EE lightweight framework," *2010 2nd International Conference on Education Technology and Computer*, Shanghai, 2010, pp. V1-442-V1-445.

[37] Q. Wu, Y. Hu and Y. Wang, "Research on Data Persistence Layer Based on Hibernate Framework," *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, Wuhan, 2010, pp. 1-4.

[38] Peng Wu and Ke Yin, "Application research on a persistent technique based on Hibernate," *Computer Design and Applications (ICCDA), 2010 International Conference on*, Qinhuangdao, 2010, pp. V1-629-V1-631.

[39] Object Oriented Analysis and Design, Benjamin Cummings Publications 1994- 2nd Edition, Grady Booch.

[40] Oriented Modeling and Design, Prentice Hall International Publication 1991, James Rumbaugh, et. al.

[41] Object Oriented Software Engineering, Addison-Wesley Professional Publication 1992, Ivar Jacobson, et. al.

[42] MoveUs, "Current Infrastructure, mobility requirements and information sources," [WWW]. Available on: http://www.moveus-project.eu/sites/default/files/moveus/files/content-files/deliverables/MovUs_D2.1%20Current%20Infraestructures%2C%20mobility%20requirements%20adn%20information%20sources_Final.pdf, pp. 122.

[43] The Unified Modeling Language [WWW]. Available on: http://www.uml-diagrams.org/uml-24-diagrams.html.

[44] Apache Derby [WWW]. Available on: https://db.apache.org/derby/index.html.

[45] ITS Factory Developer Wiki [WWW]. Available on: http://wiki.itsfactory.fi/index.php/ITS_Factory.

[46] D. Matic, D. Butorac and H. Kegalj, "Data access architecture in object oriented applications using design patterns," *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, 2004, pp. 595-598 Vol.2.

[47] Database Systems: Design, Implementation, & Management, 7th Edition, Rob & Coronel, p. 573.

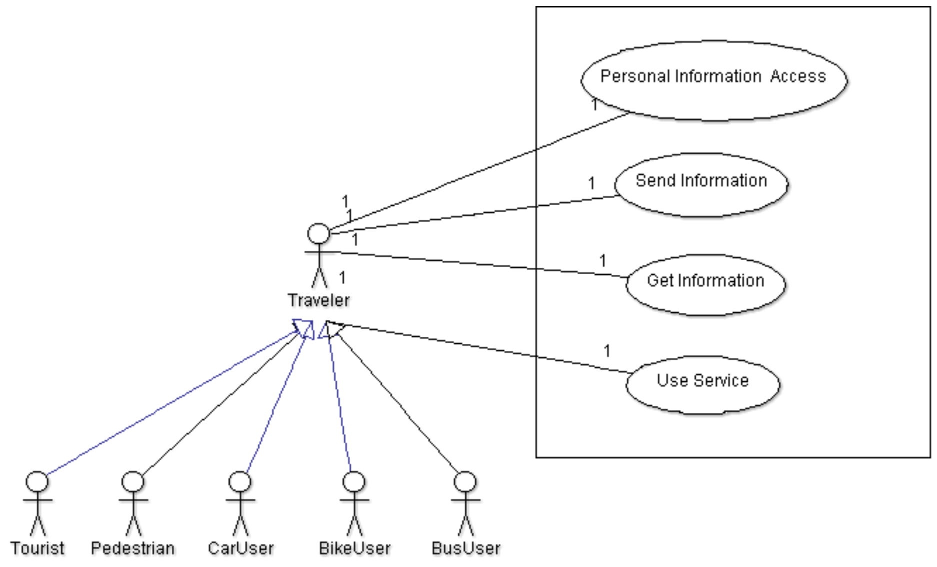
[48] Oracle J2EE Patterns – Data Access Object [WWW]. Available On: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.

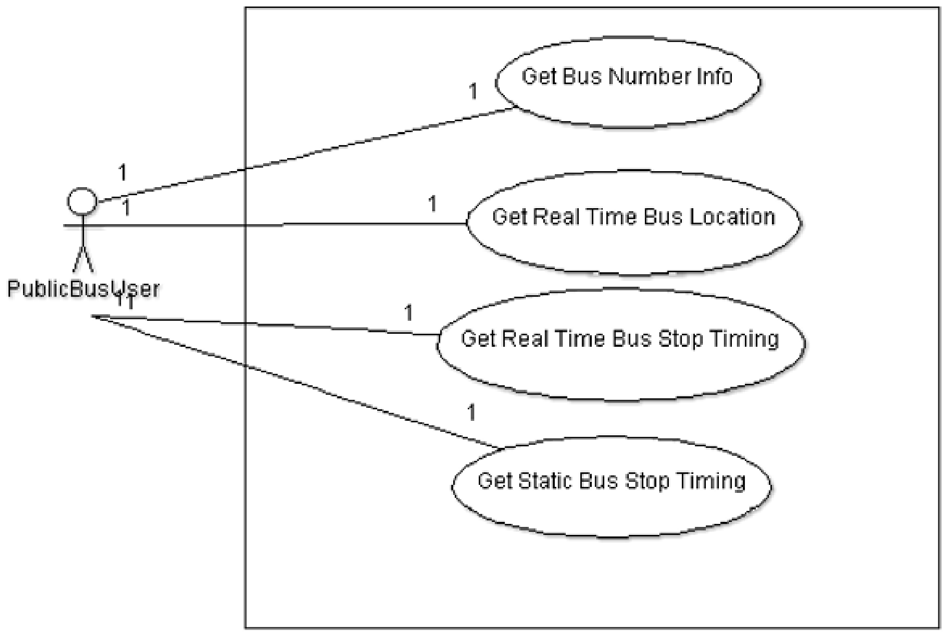
[49] D. L. Parnas. "On the criteria to be used in decomposing systems into modules," *Communication ACM* 15, Vol. 15 issue 12 Dec. 1972, pp. 1053-1058.

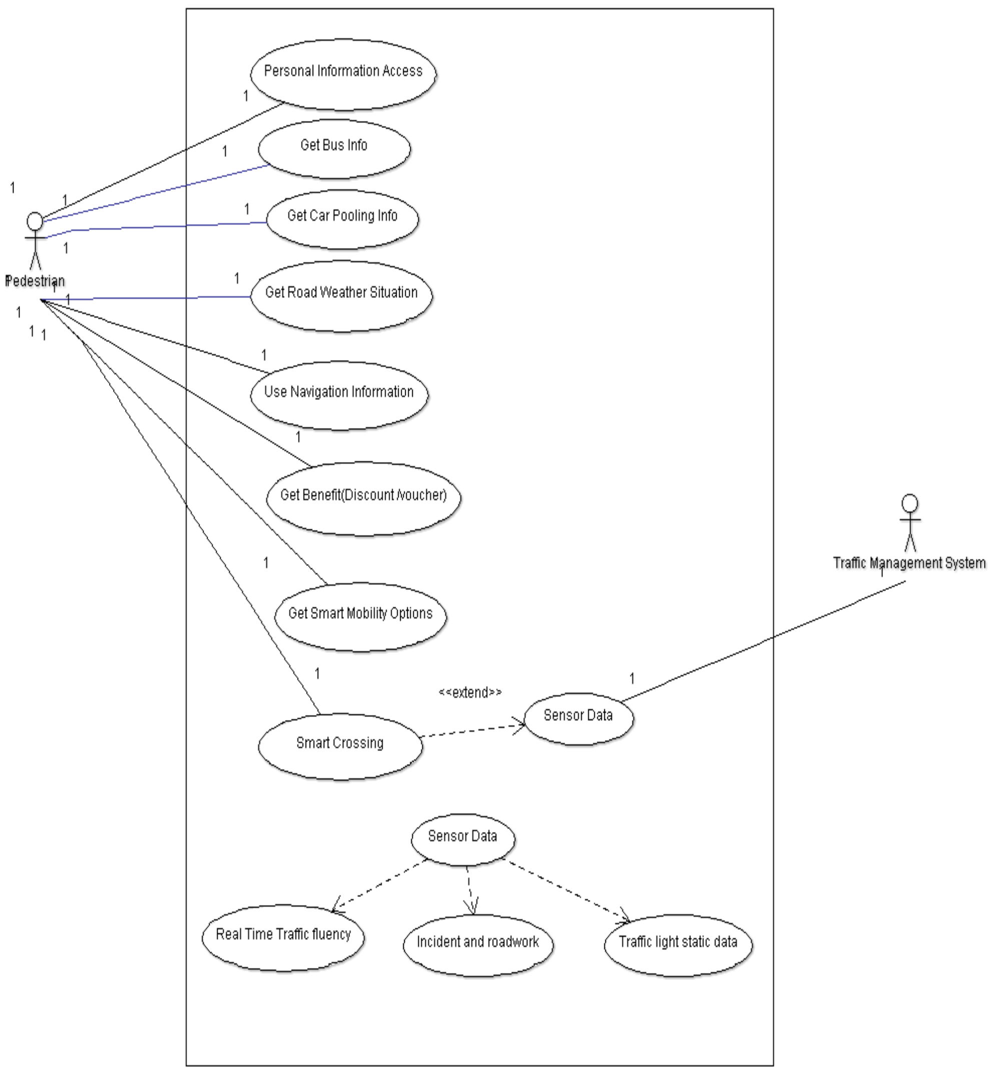
APPENDIX A: REALTIME PUBLIC BUS TESTING

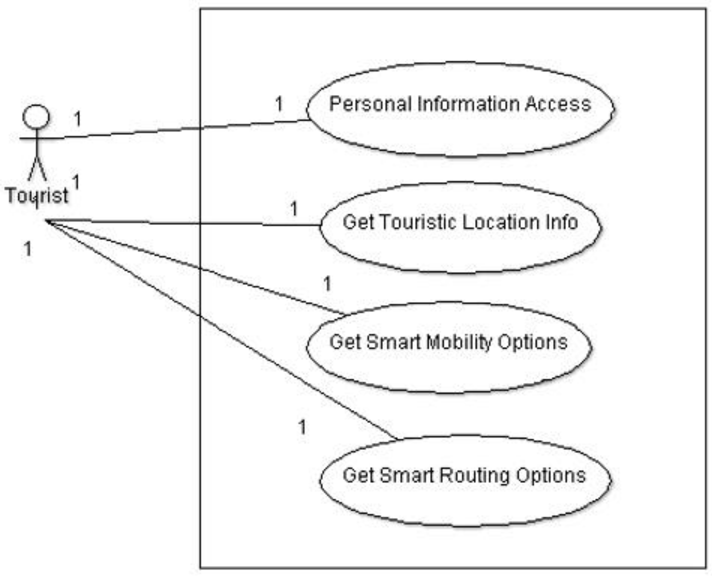
```
import static org.junit.Assert.*;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;

import dao.moveus.tut.fi.RealTimePublicBusInfoPiece;

public class RealTimePublicBusInfoPieceTest {
	private static EntityManager entityManager;

	@BeforeClass
	public static void setUpBeforeClass() throws Exception {
		EntityManagerFactory factory = Persistence
	    .createEntityManagerFactory("moveus");
		entityManager = factory.createEntityManager();
		entityManager.getTransaction().begin();
	}

	@AfterClass
	public static void tearDownAfterClass() throws Exception {
		entityManager.close();
	}

	@Test
	public void testRealTimePublicBusInfoPiece() {
		RealTimePublicBusInfoPiece realTimePublicBusInfoPiece = new
RealTimePublicBusInfoPiece();

		System.out.println("RealTimePublicBusInfoPiece" + realTimePublic-
BusInfoPiece);

		RealTimePublicBusInfoPiece foundRealTimePublicBusInfoPiece =
realTimePublicBusInfoPiece;
		System.out.println("FoundRealTimePublicBusInfoPiece" + found-
RealTimePublicBusInfoPiece);
		as-
sertEquals(foundRealTimePublicBusInfoPiece,realTimePublicBusInfoPiece);
	}

	@Test
	public void testRealTimePublicBusInfoPieceFail() {
		RealTimePublicBusInfoPiece realTimePublicBusInfoPiece = new
RealTimePublicBusInfoPiece();

		System.out.println("RealTimePublicBusInfoPiece" + realTimePublic-
BusInfoPiece);

		RealTimePublicBusInfoPiece foundRealTimePublicBusInfoPiece = new
RealTimePublicBusInfoPiece();
		System.out.println("FoundRealTimePublicBusInfoPiece" + found-
RealTimePublicBusInfoPiece);
		as-
sertEquals(foundRealTimePublicBusInfoPiece,realTimePublicBusInfoPiece);
		fail("RealTimePublicBusInfoPiece Not Found");
	}

}
```

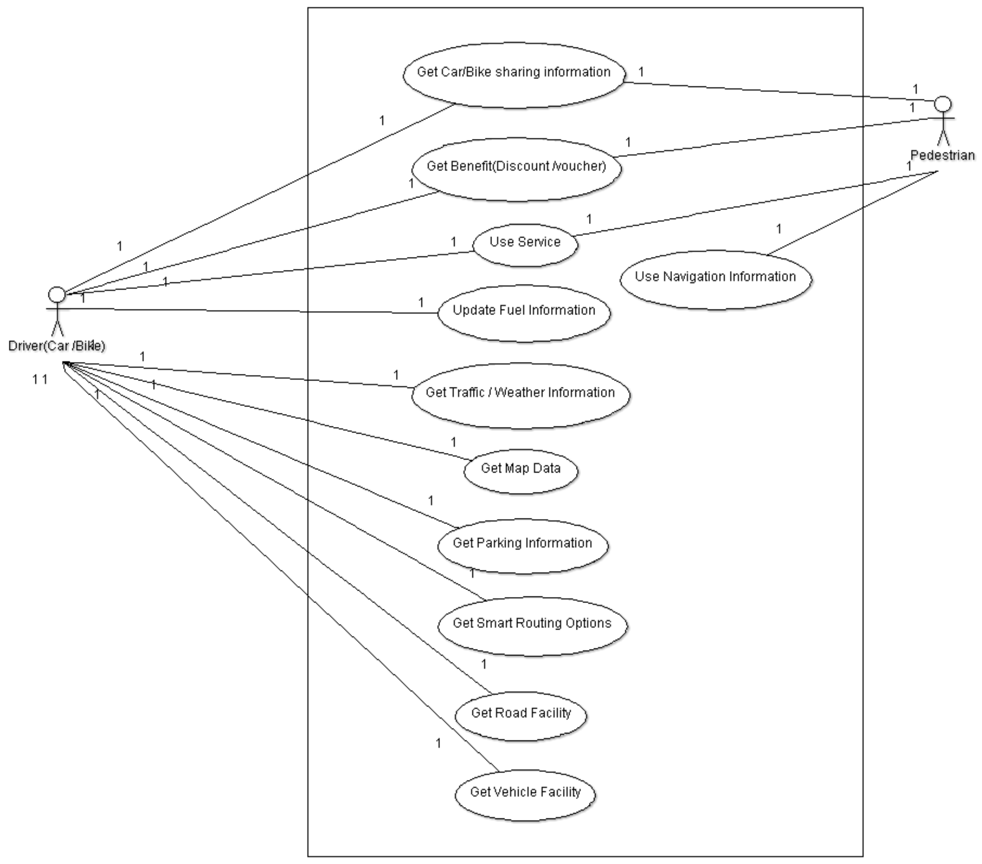
APPENDIX B: OTHER CONSIDERED USE-CASES FOR TESTING DAO




Get Bus Number Info
Get Real Time Bus Location
Get Real Time Bus Stop Timing
Get Static Bus Stop Timing
PublicBusUser


Pedestrian
Personal Information Access
Get Bus Info
Get Car Pooling Info
Get Road Weather Situation
Use Navigation Information
Get Benefit(Discount /voucher)
Get Smart Mobility Options
Smart Crossing
<<extend>>
Sensor Data
Traffic Management System
Sensor Data
Real Time Traffic fluency
Incident and roadwork
Traffic light static data


Tourist
Personal Information Access
Get Touristic Location Info
Get Smart Mobility Options
Get Smart Routing Options


Get Car/Bike sharing information
Get Benefit(Discount /voucher)
Use Service
Use Navigation Information
Update Fuel Information
Get Traffic / Weather Information
Get Map Data
Get Parking Information
Get Smart Routing Options
Get Road Facility
Get Vehicle Facility
Driver(Car /Bike)
Pedestrian

APPENDIX C: LIST OF SOME PROPOSED TEST CASES

PROPOSED TEST CASES
BikeParkingInfoTest
BikeTest
BikeUserTest
BusInfoPieceTest
BusTest
BusUserTest
CameraPresetTest
CarTest
CarUserTest
EmergencyVehicleTest
GasStationTest
IncidentsAndRoadworkTest
InfoPieceTest
JunctionWithTrafficLightLocationTest
MapInfoPieceTest
MedicalSupportLocationTest
MonitoringVehicleTest
MedicalSupportLocaitonInfoTest
MonitoringVehicleTest