IMPROVING PERFORMANCE IN VANET SYSTEM THROUGH ONTOLOGY KNOWLEDGE BASED REPOSITORY

BY

SIKIRAT KEHINDE **AINA**

DEPARTMENT OF MATHEMATICS
AHMADU BELLO UNIVERSITY, ZARIA,
NIGERIA.

MARCH, 2016.

IMPROVING PERFORMANCE IN VANET SYSTEM THROUGH ONTOLOGY KNOWLEDGE BASED REPOSITORY

BY

SIKIRAT KEHINDE **AINA**
B.Sc. Computer Science (OOU Ago-Iwoye), 2007
M.Sc./Sci./11527/2011-2012

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,
AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A
MASTER DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF MATHEMATICS
FACULTY OF SCIENCE
AHMADU BELLO UNIVERSITY, ZARIA, NIGERIA.

MARCH, 2016.

Declaration

I declare that the work in this dissertation titled "IMPROVING PERFORMANCE IN VANET SYSTEM THROUGH ONTOLOGY KNOWLEDGE BASED REPOSITORY" has been written by me in the department of Mathematics under the supervision of Dr. A. A. Obiniyi and Dr. M. A. Hammawa.

The information outlined in the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at any university.

_____ _____

Sikirat Kehinde **AINA** Date

Certification

This dissertation titled "IMPROVING PERFORMANCE IN VANET SYSTEM THROUGH ONTOLOGY KNOWLEDGE BASED REPOSITORY" by Sikirat Kehinde AINA (M.Sc./Sci./11527/2011-2012) meets the regulation governing the award of the degree of Master of Science of Ahmadu Bello University (ABU), Zaria. And it is approved for its contribution to knowledge and literary presentation.

_____	_____
Dr. A. A. Obiniyi Chairman, Supervisory Committee	Date
_____	_____
Dr. M. B. Hammawa Member, Supervisory Committee	Date
_____	_____
Prof. S. E. Adewumi External Examiner	Date
_____	_____
Prof. A. A. Tijjani Head of Department	Date
_____	_____
Prof. K. Bala Dean, School of Postgraduate Studies	Date

Dedication

This dissertation work is dedicated to Almighty Allah (SWT)! *La ilaha illa Huwa* (none has the right to be worshipped but Him).

Acknowledgement

I thank Almighty Allah, the Holy, the Creator, the most Gracious, the most Merciful, and the Wise, whose help and support are unbounded who gave me the patience and ability to reach this stage of knowledge and to my beloved Prophet Muhammad (SAW) for his teachings and guidance.

Firstly, I would like to acknowledge my supervisors Dr. A. A. Obiniyi and Dr. M. B. Hammawa for their patience, encouragement, and endless support which are appreciated beyond the limit of words.

I would like to thank my parents, my siblings and my extended families, Alhaji M. O. B. Aina, Alhaja N. A. Aina, Alhaja A. I. Aina, Alhaja B. Aina, Sis. Taiwo, Sis. Kehinde, Sis. Mariam, Sis. Fatimah, Sis. Awe, Bro. Yusuf, Sis. Sekinat, Bro. Idowu, Bro. Ayo, Sis. Billy, Taiwo (Twins sister), Sulaimon, Abdulwahid, Afusat, Habibat Raji (my niece), The Adelabu(s), The Ibrahim(s), The Olugbile(s), Prof. & Dr. (Mrs.) Taiwo, Prof. & Dr.(Mrs.) Thanni, Dr. Adeyemi, Prof. Bashar (UDU, Sokoto), Mallam Bello, Mr. Akinjide Fadeyi, and others who continuously support and encourage me. You have given me so much, and I can never pay you back. I will owe you gratitude and thankfulness forever.

I also, express my thankfulness and gratitude to other lecturers that taught me during my course of this study, Prof. S. B. Junaidu, Prof. S. Adewale and Dr. S. E. Abdullahi, also my sincere appreciation to other lecturers in the Department, the HOD Prof. A. A. Tijjani, PG coordinator Dr. A. Yahaya and others.

I would like to acknowledge the help and support that I have received from my friends and course mates during the course of my M.Sc. degree, such as Femi (class rep.), Femi Emmanuel, Laide, Firdausi, ThankGod, Rosemary, Vivian, Uncle Virus, (Yusuf, Larry and Mustapha Geophysics), Mr. CY, Member of ABU Postgraduate Muslim Sister and other.

I would also like to thank the members of Mathematics Department (such as the Librarians, Lab attendant, Office Assistant) for their valuable support.

I would like to thank my close friends Sis. Bunmi, Adedayo Akinola, Aliyu Sa'i, Alhaja Afusat & Alfa Olaide, Kofo Ayanfe (Bello), TJ, Kunle Ajayi, Niyi Dada, Tope Adesoye, Tope Macons, Mallam Nura, Rashidat Abu, Murshidat Ramadan, Fadheelah, Najib&Nusaiba Abbas, Suraj Oyewopo, Alfa Dawood, Khadijah, Fati, Ella & Christy, Rekiyah and all my well-wishers for their support and encouragement.

Finally, I would like to thank my husband Qasim Adekunle Adelabu, for his patience, support and encouragement.

May Almighty Allah grant all of you, His mercy and eternal peace (Amen).

Abstract

Vehicular Ad-hoc Network (VANET) is a technology that uses moving vehicles as nodes in a network to create a mobile network. VANET turns every participating vehicle into a wireless router or node. Several literatures reviewed have shown that information sharing among participating vehicles such as V2V (Vehicle to Vehicle) communication; V2I (Vehicle to Infrastructure) communication, lead vehicle and oppositeDirection vehicle communication have been the challenges facing VANET system. In this direction, this dissertation used ontology to develop a knowledge based system with a rule set and queries which were implemented in Java to carry out the information sharing among participating nodes in the VANET system. The proposed improvement has been illustrated from the comparison table and graph provided in the chapter four of this dissertation such that communication was created between Autonomous vehicles, human driven vehicles and OppositeDirection vehicles.

Table of Contents

Cover Page ..... **I**

Title Page ..... **II**

Declaration ..... **III**

Certification ..... **IV**

Dedication ..... **V**

Acknowledgement ..... **VI**

Abstract ..... **VII**

Table of Contents ..... **VIII**

List of Tables ..... **XI**

List of Figures ..... **XII**

Abbreviations ..... **XV**

1.0 INTRODUCTION **1**

1.1 Background of Study 1

1.1.1 Vehicular ad hoc network (VANETs) 2

1.1.2 Ontology 3

1.2 Research Problem 3

1.3 Motivation for the Research 6

1.4 Aim and Objectives 6

1.5 Research Methodology 6

1.6 Contribution to Knowledge 7

1.7 Dissertation Organization 7

2.0 LITERATURE REVIEW **8**

2.1 Ontology and Ontologies Language 8

2.1.1 Ontology definitions 8

2.1.2 Ontology language 9
2.1.3 Ontology tool 13
2.1.4 Standard ontology query languages and rule languages 14
2.1.5 Ontology reasoners 16
2.2 Introduction to Vehicular Ad hoc Network (VANET) 20
2.2.1 Basic entities of the network 21
2.2.2 Basic Communication Entities 22
2.2.3 Types of Wireless Network 23
2.2.4 Standards VANET communications 23
2.3 Related Works 24
2.4 Existing System 32
3.0 ONTOLOGY DEVELOPMENT AND RULE SYSTEM FOR VANET33
3.1 Introduction 33
3.2 System Architecture of Semantic Web Based VANET 33
3.3 Building a Knowledge Base for VANET 36
3.3.1 Ontology Development 36
3.3.2 VANETs Ontology Concepts Enumeration 37
3.3.3 VANET Domain Knowledge 38
3.4 A Protocol for Dynamically Populating VANET Ontology 41
3.5 Event Control in VANET through Rule System 42
3.5.1 Basic interoperability of SWRL and OWL 42
3.5.2 Proposed VANET rule set 43
4.0 IMPLEMENTATION AND DISCUSSION 45
4.1 Introduction 45
4.2 Implementation Tools: Protégé, NetBeans and Jess 45

4.3 Coupling of Rule Engine with Development Environment (NetBeans)..... 46
4.5 The Semantic Web Based VANET Application..... 51
4.5.1 The two – fold panels..... 51
4.5.2 Control panels key activation 52
4.6 Comparison with closest Work..... 59
4.7 Discussion..... 60
5.0 SUMMARY, CONCLUSION AND RECOMMENDATION..... 62
5.1 Summary..... 62
5.2 Conclusion 62
5.3 Recommendation for Future Work 63
REFERENCES..... 64
APPENDIX 70

List of Tables

Table 2.1: Types of Wireless Networks..... 23

Table 2.2: Limitations Morignot and Nashashibi, (2012) 27

Table 2.3: Limitations Armand et al. (2014) 31

Table 3.1 VANETs System Functionality Question..... 43

Table 4.1: Comparison of Proposed System with some Previous Closest Work 59

Table 4.2: Comparison of Proposed System with some Previous Closest Work Modified 59

List of Figures

Figure 1.1 The Road Condition in one of the major commercial area in Lagos Nigeria (Williams, 2013) 5

Figure 2.1 Main Components of the Pellet Reasoner (Sirin *et al*., 2007). 17

Figure 2.2 RacerPro System Architecture (Abburu, 2012). 18

Figure 2.3 Main Software Modules of ELK and Information Flow during Classification (Kazakov *et al*., 2012). 19

Figure 2.4 Jess Architecture Diagram (Abburu, 2012). 20

Figure 2.5 A Typical VANET Scenario (Kao, 2004) 21

Figure 2.6 Architecture of Intelligent Driver Assistance System (Saravanan *et al*., 2010) 24

Figure 2.7 High-level Ontology Representation Model (Morignot and Nashashibi, 2012) 26

Figure 2.8 Ontology for Situation Assessment (Pollard *et al*., 2013) 28

Figure 2.9 High Level Ontology (Erritali *et al*., 2013) 29

Figure 2.10 Ontology Classes, Object and Data Properties (Armand *et al*., 2014) 30

Figure 2.11 Ontology Development Process (Boyce and Pahl, 2007) 32

Figure 3.1 System Architecture 33

Figure 3.2 Class Hierarchy 39

Figure 3.3 Creating ObjectProperty 39

Figure 3.4 Class Creation 39

Figure 3.5 Adding Instances to Ontology 39

Figure 3.6 Object Property 40

Figure 3.7 Individual Members List 40

Figure 3.8 The Ontology Designed 40
Figure 4.1 Package Listing of the Java Files 46
Figure 4.2 Navigate the NetBeans Section for Adding Libraries to a Project 46
Figure 4.3 Listing of the Jess Rule Engine jar Files 47
Figure 4.4 Classes and Packages of the Jess Rule Engine Imported into the Java File 47
Figure 4.5 Semantic Rule Set for the VANET System 48
Figure 4.6 Vehicle Ontology File Opened in Netbeans Environment 48
Figure 4.7 Rule Set in Development Environment 49
Figure 4.8 Development of the GUI in Java 49
Figure 4.9 Development of the GUI in Java 50
Figure 4.10 Rule Set Query Implementation in Java 50
Figure 4.11 Two – Fold Panels GUI 52
Figure 4.12 Outcome of Key Activation in the Two Fold Panels GUI showing Ontology Uploaded 53
Figure 4.13 Outcome of Key Activation in the Two Fold Panels GUI showing Choice of Lane to Insert Vehicle 53
Figure 4.14 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted 54
Figure 4.15 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (A) Applied 54
Figure 4.16 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (B) Applied 55
Figure 4.17 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (C) Applied 55

Figure 4.18 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (D) Applied 56
Figure 4.19 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (E) Applied 56
Figure 4.20 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (F) Applied 57
Figure 4.21 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted into Right Lane 57
Figure 4.22 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted into Left and Right Lanes 58
Figure 4.23 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted into Left and Right Lanes 58
Figure 4.24 Comparison of the Proposed System with Some Previous Closest Work 60

Abbreviations

ADAS: Advance Driving Assistance System

ASTM: American Society for Testing and Materials

CML: Conceptual Modeling Language

DAML+OIL: DARPA Agent Markup Language and Ontology Inference Layer

DHCP: Dynamic Host Configuration Protocol

DLP: Description Logic Programs

DSRC: Dedicated short-range communication

DTD: Document Type Declaration

EO: Ontology Enterprise

FaCT: Fast Classification of Terminologies

GPS: Global Positioning System

InVANET: Intelligent Vehicular ad-hoc network

IP: Internet Protocol

ITS: Intelligent Transportation System

JESS: Java Expert System Shell

KIF: Knowledge Interchange Format

MANETs: Mobile Ad Hoc Networks

OBU: On Board Unit

OWL: Web Ontology Language

RACER: Renamed ABoxes and Concept Expression Reasoner

RDF: Resource Description Framework

RDFS: Resource Description Framework Schema

RSU: Road Side Unit

RVC: Roadside-to-vehicle communication

SPARQL:	Simple Protocol and RDF (Resource Description Framework) Query Language
SQWRL:	Semantic Query -enhanced Web Rule Language.
SWQL:	Semantic Web Query Language.
SWRL	Semantic Web Rule Language
TOVE:	TOronto Virtual Enterprise
V2I:	Vehicle-to-Infrastructure
V2V:	Vehicle-to-Vehicle
VANETs:	Vehicular Ad-hoc Networks
WAVE:	Wireless Access in Vehicular Environment
XML:	Extensible Markup Language

CHAPTER ONE

INTRODUCTION

1.1 Background of Study

The concept of networking in computing became necessary even as information sharing among group of users, either on intra or inter relational basis began to affect the communication pattern among people from all walks of life. Majorly, what took toll then was simple networking of computer to computer, popularly called point-to-point connection. Advances in research birthed several networking topologies which includes ring, mesh, star, bus topologies and a host of others. Networking among devices has been categorized into Local Area Network (LAN) and Wide Area Network (WAN). Technically, these networking patterns described previously have been stationed. Stationed in the sense that, each participating node (machine) in the network does not move or displace itself from one network to another network even while it carries out its network related task, say file sharing. Hence, a network administrator configures the network with this notion in view. Furthermore, machines participating in networks were dynamically assigned Internet protocol (IP) address. This makes connectivity quite liberal, providing intending machines the privilege of attaching and detaching themselves to the network at any time.

Hence, node enters and reenters a network several times and at the same time utilizing different IP addresses. But, since the manual IP configuration was seemingly left out of the hands of the network administrator, there arises the need for a dynamic host configuration protocol (DHCP) server which dynamically assigns IP address to machines attempting connectivity to the network and as well reclaiming the IP address of machines that are disconnecting themselves from the network.

Technological advances began to give birth to new and better shape to networking. Mobile devices and other electronically driven machines having the tendency of being moved from one point to another even while they perform their functions began to demonstrate the need for network connectivity even while in motion. This necessitated more research that brought forth the concept of mobile ad-hoc network (MANET) and vehicular ad-hoc network (VANET).

This network pattern allows host or nodes to connect and disconnect to and from the system respectively at any time, though they continue to carry out their network aided tasks while reconnecting to other network. The word ad-hoc intends to point out that each node in the network is not permanent but the nodes coupled or networked together to quickly and jointly perform a task afterward will be disconnected. According to Tarik *et al*., (2007) and Shastri *et al*., (2011) ad-hoc network is said to be an infrastructureless network with no fixed routers, hosts or wireless base stations. Vehicular Ad-hoc Networks (VANETs) represent a rapidly emerging, particularly challenging class of Mobile Ad Hoc Networks (MANETs).

1.1.1 Vehicular ad hoc network (VANETs)

A vehicular ad hoc network (VANETs) is an application of mobile network where every participating vehicle communicates with each other (peer to peer manner) and as well serve as a wireless router (Shastri *et al*., 2011).

VANETs are known to be a distributed, self-organizing communication network built up by moving cars as nodes in a network to create a very high mobility of nodes and limited degrees of freedom in nodes movement patterns. It turns every participating car

into a wireless router or node, allowing cars approximately 100 to 300 meters from each other to connect and in turn, create a network with a wide range.

VANETs are developed for enhancing the driving safety and comfort of automotive users. The VANETs can provide wide variety of services such as Intelligent Transportation System (ITS), Intelligent Vehicular ad hoc network (INVANET), Dedicated short-range communication (DSRC), Roadside-to-vehicle communication (RVC) e.g. safety applications (Kirthiga and Karthik, 2014).

VANETs has gained an important part of the interest of researchers and become very popular. More specifically, VANETs operate without fixed infrastructure and can survive rapid changes in the network topology (Shastri *et al*., 2011).

1.1.2 Ontology

In philosophy, ontology is the study of being and existence. It is often said that ontologies "carve the world at its joints." Ontologies according to Mihoubi *et al*., (2000), are used in order to support interoperability and common understanding between different parties and are key components in solving the problem of semantic heterogeneity, thus enabling semantic interoperability between different web applications and services.

Ontologies provide a common understanding of a domain that can be communicated between people of heterogeneous and widely spread application systems. Ontology has been used in several domain of computer science even in ad-hoc network system to find solution, such as vulnerability in security aspect (Verma and Gujral, 2012).

1.2 Research Problem

VANETs have been categorically deployed into vehicular network in addressing the mobility of each vehicle using such facility as in vulnerability of VANET issues.

Majorly, congestion and road network maneuvering by vehicles is a challenge to be combated in traffic management. Several transport management systems have been deployed to manage passenger- vehicle interactivity, and these have to an extent circumvent some of the problems encountered in those systems. Autonomous vehicles have also been researched into, though still seeking legislative backing (in United State) to move on the road with human driven vehicles, and built, tested and proven road worthy. However, what remains a challenge is information sharing platform among moving vehicle on the road. The idea is whether a vehicle is human driven or autonomous, there should be an information exchange in tackling road meandering between vehicles. Interestingly, the pivotal challenge to be addressed by VANETs is the issue of road congestion (Zhu and Roy, 2003). Clamor for employment among employable graduates and a desire after better social and infrastructural facilities by youths had positioned most of the urban areas – cities and metropolitans – to high influx of people. Moreover, nations with fast economic growth have not been spared of an uncontrolled increment in car usage on their roads. This put the roads on serious congestions and with ever increasing accident rate among vehicles struggling for the already congested road. Figure 1.1 captures some horrible congestion experienced on some major roads. Hence, VANETs provide a networking medium where vehicles can exchange information about what they have encountered on the road so that another vehicle coming in the opposite direction or moving towards a point where there is either congestion or accident can decide to take another road.

Another benefit of VANET is to help vehicle navigate their way through shortest routes. This saves both time and vehicle resources, especially in a city or settlement where both working class and business men reside and time is considered a great factor among them.

VANET has data repository for each vehicle and that each vehicle has to intelligently reason or make inference over the information gathered by another vehicle in other to be able to mine both helpful and instructive information from the ad-hoc network. Hence, one of the challenges encountered in VANET is the data representation or modeling, so that the smart agents' vehicle can intelligently mine reasonable information out accurately with high precision (Tarik *et al*., 2007).



Figure 1.1 The Road Condition in one of the major commercial area in Lagos Nigeria (Williams, 2013)

1.3 Motivation for the Research

The Nigeria traffic conditions for instance, Figure 1.1 denotes a typical situation of road condition in Nigeria. Research on several technologies of Advance Driving Assistance System (ADAS), has shown how it helps in transportation system in foreign countries. It is realized that if ADAS technology can be improved upon, the technology can also be incorporated into human driven vehicles to aid the interaction and communication among vehicles and multiagent transportation system.

1.4 Aim and Objectives

The aim of this dissertation is to improve performance in VANET system through the development and implementation of an enriched VANET ontology knowledge base.

The objectives of this dissertation are to:

a. Develop ontology-based and dynamic VANET knowledge base.
b. Create rule set (inference) that provide an enhanced reasoning capability.
c. Implement a communication channel between VANET and its ontology knowledge base.

1.5 Research Methodology

The aim and objectives of this dissertation were achieved by:

a. Review of literature on ADAS technologies, ontology and semantic.
b. Protégé 3.4.8 & 5.0 were used to carry out the ontological development.
c. Java: was used to implement the ontology designed in connection with the rule engine.
d. SWRL: was used to add rules to the ontology.
e. SQWRL: was used to query the OWL ontology.
f. Comparison of the proposed system and the previous related work.

1.6 Contribution to Knowledge

a) A rule set (inference) was created that provided an enhanced reasoning capability for the ontology knowledge based.
b) The study has successfully achieved the communication system which involved vehicle in OppositeDirection.
c) A system usable for both Autonomous vehicle and human driven vehicle was implemented.

1.7 Dissertation Organization

This dissertation is organised as follows:

Chapter 1: General introduction to VANET and ontology, motivation for the research, aim and objectives, research methodology and contribution to knowledge were discussed.

Chapter 2: Literature Review which gives detail about concept of ontology, VANET and related works.

Chapter 3: Presents the detail of the ontology development and rule system for VANET.

Chapter 4: Implements the system architecture presented in chapter three, and then discusses the result of the implementation in a bid to put forward the realization of the aim and objectives of this dissertation.

Chapter 5: The research summary, conclusion and recommendation for future work were given.

CHAPTER TWO

LITERATURE REVIEW

2.1 Ontology and Ontologies Language

Ontology is a knowledge base that has given resolution to many aspects of computer science, its related fields and their applications.

2.1.1 Ontology definitions

Ontology which is taken as the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base (Bernaras *et al.,* 1996*).* It is also defines as a hierarchically structured set of terms for relating a domain that can be used as a skeletal basis for a knowledge base (Swartout *et al.*, 1997*).*

Ontology also defines basic terms and relations comprising the vocabulary of a topic area, as well as the rules for linking terms and relations to define extensions to the vocabulary (Neches *et al.*, 1991*).* It is being defined as an explicit, formal specification of a shared conceptualization of a domain of interest (Gruber, 1993; Studer *et al.* ,1998; Seddiqui and Aono, 2010), which implies that it is a knowledge base which enables devices to communicate with each other effectively. Ontology can also be defined as a system that permits its designers to represent concepts in a given domain, and further show the relationships among these concepts (Obiniyi *et al*., 2014). Ontology is abbreviated as FESC (Formal, Explicit, and Specification of Shared Conceptualization) (Jain and Singh, 2013; Bello, 2014). Where the foundations of this definition are:

a) **Formal**: It specifies that it should be machine understandable.

b) **Explicit**: It defines type of constraints used in model.

c) **Shared**: It means that Ontology is shared by the groups. It is not restricted to individuals.

d) **Conceptualization**: It refers to the model of some phenomenon to identify relevant concepts of that phenomenon.

Ontology is nowadays popular in many varieties of applications in given ontological definition or vocabulary for a researcher or programmer that needs to share information in a domain. Ontology has hierarchical level or generates a level of abstraction that largely characterizes a model and the dependencies of activities Nair and Dutta, (2010) and consequently being valuable for system to interact and helpful in the development process. (Verma and Gujral 2012).

2.1.2 Ontology language

Ontologies are used to capture knowledge about some domain of interest. Ontology describes the concepts in the domain and also the relationships that hold between those concepts. Different ontology languages provide different facilities.

Ontologies are developed using specific languages, such as XML, Knowledge Interchange Format (KIF) and OWL (Wooldridge, 2009) and others like RDF, RDFS, and DAML+OIL.

a. Resource Description Framework (RDF)/RDFS

RDF(S) was developed to represent information in the Web. In RDF(S) the resources in the Web are identified by Web identifiers (Uniform Resource Identifier or URI) (Manola and Miller, 2004). To make it machine processible, RDF inherits XML-based syntax. After many years of development, RDF(S) now has formal syntax, formal semantics and XML Schema datatypes. It is a W3C recommended information representation standard for the Semantic Web. The RDF abstract syntax has a graph

pattern, where the statements are represented as N-triples (format: Subject – Predicate – Object or node-arc-node link, hence the term ‘graph’) (McBride, 2004; Klyne and Carroll, 2004). RDF can express resource properties and their values. RDFS extends RDF by providing the ability to define RDF vocabularies such as classes, properties, types, ranges and domains. However, RDF and RDFS only have very limited expressive powers (Manola and Miller, 2004; Ratnakar and Gil, 2002):

i. RDF(S) cannot express equality and inequality;
ii. RDF(S) cannot define enumeration of property values;
iii. RDF(S) cannot apply cardinality and existence constraints;
iv. RDF(S) cannot describe unique, symmetric, transitive, inverse relationships among properties;
v. RDF(S) cannot describe union, intersection and complement;
vi. The domain and range in RDF(S) can only be specified globally Ratnakar and Gil, (2002).

As a result, several more sophisticated languages have been developed to meet these 6 requirements.

b. DARPA Agent Markup Language (DAML) and Ontology Inference Layer (OIL)

According to Horrocks, (2002) DAML+OIL is a combination of DAML (DARPA Agent Markup Language) and OIL (Ontology Inference Layer). Fensel *et al*., (2001) stated that it has an RDF/XML syntax based on the frame paradigm and describes the structure of a domain (schema) in an object-oriented style. DAML+OIL consist of a set of axioms asserting the relationships between classes and properties.

DAML+OIL use a Description Logic style model theory to formalize the meaning of the language (Horrocks *et al*., 2003). This is a very important feature to reduce arguments and confusions, thus giving the language the ability to precisely represent the meanings of information (Zhang, 2005). This ability is crucial for automatic reasoning, which is the goal of the Semantic Web. The new features DAML+OIL supports are the following:

i. Constraints (restrictions) on properties (existential/universal and cardinality),
ii. Boolean combinations of classes and restrictions, e.g., union, complement and intersection,
iii. Equivalence and disjointness,
iv. Necessary and sufficient conditions, and
v. Constraints on properties.

Nevertheless, DAML+OIL try to be compatible with RDF syntax, but this raises some serious syntactic and semantic problems (Horrocks *et al*., 2003). Another problem is that DAML+OIL datatypes are not compatible with RDF, since RDF did not provide datatype definition ability when DAML+OIL were being developed.

c. Ontology Web Language (OWL)

According to Smith *et al*., (2004) OWL was developed on top of RDF and borrowed from DAML+OIL. Like RDF, OWL is the standard recommended by W3C for Semantic Web. OWL is powerful in expression, but complex for computation. To compromise between expressive power and acceptable computational complexity, OWL has three increasingly-expressive sublanguages:

Ontology web language Lite (OWL Lite), OWL Description Language (DL), and OWL Full. Among them, OWL Lite is a subset of OWL DL, and OWL DL is a subset of

OWL Full. OWL Full contains the entire OWL language constructs and provides the free, unconstrained use of RDF constructs (Dean, 2004; Pandey, 2012).

In OWL Full, *owl:Class* is equivalent to *rdfs:Class*. OWL Full also permit classes to be individuals. A class can even be a property of itself. In OWL Full, *owl:Things* and *rdfs:Resource* are equivalent too. This means that object properties and datatype properties are not disjoint.

d. Ontology Web Language 2 (OWL2)

It consists of three sublanguages such as OWL2-RL (Rule Language), OWL2-QL (Query Language) and OWL2-EL (Obiniyi *et al*., 2014). OWL 2 is an improved version of OWL 1 to make ontology development stress-free and richer. OWL 2 ontologies are also called RDF graph. Most OWL 2 tools support its exchange in RDF/XML format and its corresponding serialization formats. This has two different ways of reading meaning into its ontologies. That is Direct Semantics and RDF-Based Semantics, and these two different semantics are utilized by semantic web tools and reasoners' requests or queries. While the Direct Semantics provides direct meaning into ontology structure, the RDF-Based Semantics provides reading meaning into the RDF graph. However, there exists a link between these two semantics of OWL 2 which states thus: inferences generated from an OWL 2 DL using Direct Semantics will still be correct if the resulting ontology is mapped into an RDF graph and the graph is read using the RDF Semantics (Obiniyi *et al*., 2014).

OWL 2 QL is used for applications whose underlying ontology is made up of large number of instances that will be queried, and this reduces expressivity of the profile and it also allows information stored on a relational database to be queried beside querying its own ontology. OWL 2 RL is mostly used in conjunction with rule language, and this

enhances expressivity for RDF based applications. Its reasoning tasks are implemented using some rule engines. For OWL EL, the profile is based on the εL family of description logics that provide only existential (and no universal) quantification. It is used for applications that basically need high number of properties and classes, and it provides sufficient expressive power for ontologies that use it (Obiniyi *et al.*, 2014).

2.1.3 Ontology tool

Ontology editors are tools that enable the users for inspecting, browsing, codifying, and modifying ontology and support in this way the ontology development and maintenance task (Sure, 2002). The existing editors vary in their complexities such as the underlying knowledge model, usability and scalability. Nevertheless, all of them provide enough support for the initial ontology development. Prot´eg´e project, OntoEdit Sure *et al*., (2002), OilEd Bechhofer *et al*., (2001), WebODE Arp´ırez *et al*., (2001) and Ontolingua Farquhar *et al*., (1997) are some examples of ontology editors. In this dissertation a brief discussion will be done on protégé since it is the editor chosen for this work because of its flexibility and its ability to extend with other applications.

a. Protégé

Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. Further, Protégé can be extended by way of a plug-in Architecture and a Java-based Application Programming Interface for building knowledge-based tools and applications.

b. Protégé editor: Protégé-OWL editor permits the users to Load and save OWL and RDF ontologies, Edit and visualize classes, properties, and SWRL rules, define logical class characteristics as OWL expressions, execute reasoners such as description logic classifiers and Edit OWL individuals for Semantic Web markup.

2.1.4 Standard ontology query languages and rule languages

Based on the ontology languages described previously, several query languages and systems have been developed.

a. Simple Protocol and RDF (Resource Description Framework) Query Language (SPARQL)

SPARQL stands for SPARQL Protocol and RDF Query Language. SPARQL according to Seaborne and Prud'hommeaux, (2005), is a Server-Client-based RDF query language. It has SQL syntax and is influenced by RDQL and SquishQL (http://swordfish.rdfweb.org/rdfquery/). SPARQL supports disjunction in the query and thus can process more complex query than RDQL. SPARQL also provides optional variable binding and result size control mechanisms for real world usage.

b. Semantic Web Rule Language (SWRL)

Semantic Web Rule Language, an acronym for SWRL, is a rule language. SWRL (SWRL, www.w3.org/Submission/SWRL/) is defined as language combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/ Binary Datalog). The specification was submitted to W3C in May 2004 by the National Research Council of Canada, Network Inference (since acquired by web Methods), and Stanford University in association with the Joint US/EU ad-hoc Agent Markup Language Committee. Compared to Description Logic Programs (DLP), a slightly earlier proposal for integrating description logic and Horn rule formalisms by an overlapping authoring team, SWRL takes the opposite integration approach: DLP can be seen as the 'intersection' of description logic and Horn logic; SWRL, as roughly their 'union'. For DLP, the resulting rather inexpressive language corresponds to a peculiar looking description logic imitating special rules. It is hard to see the DLP restrictions, which stem from Lloyd-Topor transformations, being either

natural or satisfying. On the other hand, SWRL retains the full power of OWL DL, but adds rules at the price of undecidability and a lack of complete implementations, although the SWRL Tab of Protégé has become quite popular (Hu *et al*., 2009). Rules in SWRL are of the form of an implication between an antecedent (body) conjunction and a consequent (head) conjunction, where description logic expressions can occur on both sides. The intended interpretation is as in classical first-order logic: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

c. Semantic Query-enhanced Web Rule Language (SQWRL)

SQWRL is an extension of SWRL rule language (SWRL-Submission, http://www.w3.org/submission/SWRL). It takes rules' antecedent as a pattern specification for a query and takes rules' consequent as a retrieval specification. Any valid SWRL antecedent is a valid SQWRL pattern specification. That means, SQWRL places no restriction on the left side of a query. It uses SWRL's built-in libraries as an extension point (SWRL-Submission, http://www.w3.org/submission/SWRL). Also SWRL editor can be used to generate and edit the queries but to execute the SQTab has to be included by adding the jar file JESS 7.1p2 in the protégé plug-in directory and the core operator is aqwrl:select. The select operator takes one or more arguments, which are variables in the pattern specification of the query, and builds a table using the arguments as the columns of the table. The built-ins like, sqwrl:count, sqwrl:orderBy, sqwrl:avg can be used as in SQL. SQWRL does not support subqueries, but it is achieved by using the intermediate inferences made by SWRL rules. This mechanism is used to decompose the complex queries. The sqwrl:makeSet built-in is used to create a set. Using this set and sqwrl built-ins like, union, difference, isEmpty, size, and

intersection are used to do set operations. The SQWRL query tab provides a graphical interface to display the results of SQWRL queries (Mohan and Arumugam, 2011).

2.1.5 Ontology reasoners

A reasoner is a program that infers logical consequences from a set of explicitly asserted facts or axioms and typically provides automated support for reasoning tasks such as classification, debugging and querying (Dentler, 2011).

A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms (Abburu, 2012).

To ensure the quality of ontologies, there is a need for dealing with the inconsistency and uncertainty in the ontologies of real-world applications.

An inconsistent ontology means that an error or a conflict exist in an ontology, as a result some concepts in the ontology cannot be interpreted correctly. The inconsistency will result in false semantic understanding and knowledge representation. An uncertain ontology means that the correctness of the ontology is probabilistic. Ontology reasoning reduces the redundancy of information in knowledge base and finds the conflicts in knowledge content. The inference rules are commonly specified by means of a description language. Many reasoners use first-order predicate logic to perform reasoning; inference commonly proceeds by forward chaining and backward chaining. Forward chaining and backward chaining are the strategies of ontology reasoners (Kiryakov and Damova, 2011).

a. **Forward-chaining:** According to this strategy the reasoner starts from the known facts and derive valid inferences. The goals of such reasoning are:

   i. To compute the inferred closure.

ii. To answer a particular query.

iii. To infer a particular sort of knowledge (e.g., the class taxonomy).

b. Backward-chaining: According to this strategy the reasoner starts from a particular fact or a query and to verify it or to find all possible solutions.

Mishra and Kumar, (2010) presented an extensive survey of nineteen reasoners that have been released between 1975 and 2009.

Among the large number of reasoners available the popular reasoners and suited for protégé and other editor toolkit are many, in this dissertation, few of it will be discussed: Pellet, RACER, FACT++, HermiT, ETK and JESS. See more from (Mishra and Kumar, 2010; Abburu, 2012).

a. Pellet

Pellet is an open source java based OWL-DL reasoner developed by The Mind Swap group. It is based on the tableau algorithm and supports expressive description logics. It is the first reasoner that supported all of OWL DL SHOIN(D) and has been extended to OWL2 (SROIQ(D)) (Sirin *et al*., 2007). Pellet supports OWL 2 profiles. It reasons ontologies through Jena as well as OWL-API interfaces. Figure 2.1 shows various components of the pellet reasoner.

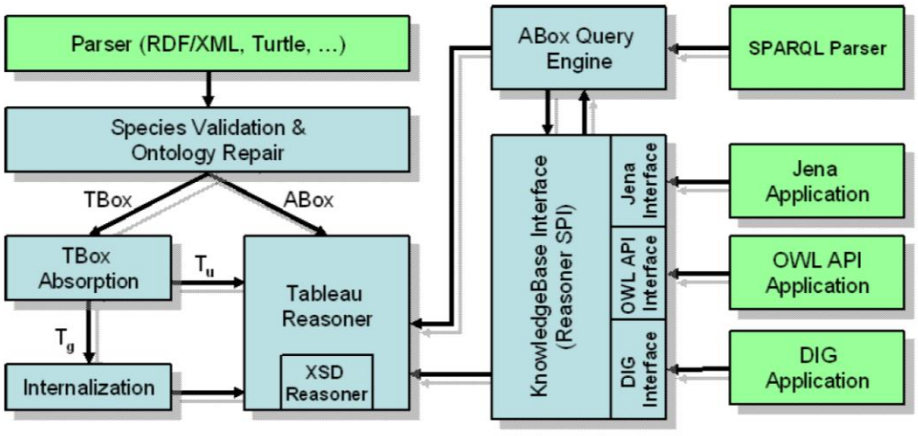


Figure 2.1 Main Components of the Pellet Reasoner (Sirin *et al*., 2007).

b. Renamed Aboxes and Concept Expression Reasoner (RACER)

Horrocks *et al*., (2003) developed a reasoning model known as RACER (Renamed ABoxes and Concept Expression Reasoner). RACER, also known as RacerPro by Haarslev *et al*., (2011) is the first OWL reasoner. It supports the optimization techniques of FaCT as well as the new optimization techniques for dealing with number restrictions and ABoxes. RACER implements TBox and ABox reasoner for the SHIQ logic. Figure 2.2 shows clear architecture of the reasoner RacerPro.

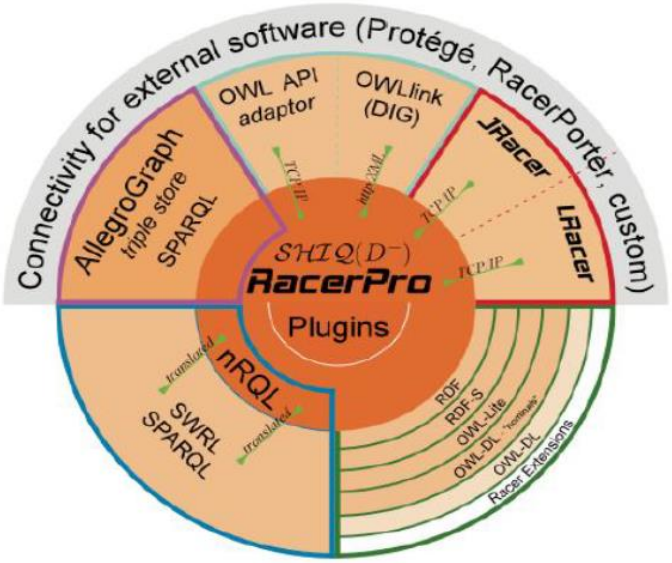


Figure 2.2 RacerPro System Architecture (Abburu, 2012).

c. Fast Classification of Terminologies (FaCT ++)

Horrocks has presented a reasoner known as FaCT (Fast Classification of Terminologies). It can be used as a description logic classifier and for modal logic satisfiability testing. The FaCT system has sound and complete tableaux algorithm for expressive description logics. An updated version of FaCT is FaCT++. This reasoner uses the same algorithm as in FaCT, but with a different internal structure. It is implemented in C++. The first version of the FaCT++ only supported the reasoning in SHOIQ, OWL-DL. However, the latest version of FaCT++ supports OWL and is based

on the description logic SROIQ. FaCT++ implements a tableau-based decision procedure for general TBoxes and incomplete support for ABoxes (Sattle, 2007).

d. HermiT

HermiT is the first publicly available OWL reasoner. It is written using OWL. HermiT can check the OWL files to determine the consistency of the ontologies and to identify the hierarchical relationships between the classes. This reasoner is based on hypertableau calculus, the calculus which addresses performance problems due to nondeterminism and model size. It also provides the faster process for classifying the ontologies (Abburu, 2012; Shearer *et al*., 2008).

e. ELK

ELK is a flexible system which allows variety of configurations. It is supported by a modular program structure that is organized using the Apache Maven build manager for Java. The ELK reasoner is also a free and open source reasoner for the lightweight ontology language OWL 2 ELK runs in all operating systems that support Java 1.5 or above. Figure 2.3 shows Main software modules of ELK reasoner and information flow during classification (Kazakov *et al*., 2012)

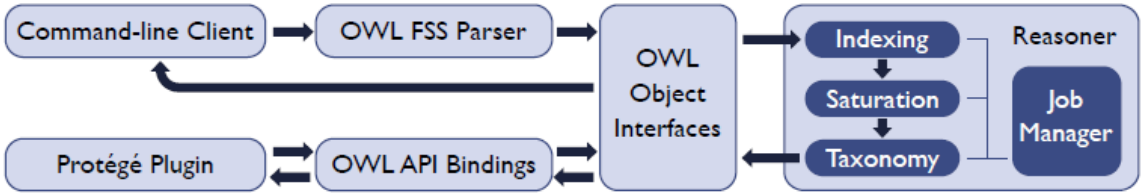


Figure 2.3 Main Software Modules of ELK and Information Flow during Classification (Kazakov *et al*., 2012).

f. Java Expert System Shell (JESS)

Java Expert System Shell (JESS) was originated from AI research in the 70s and 80s and was developed at Sandia National Laboratories in late 1990s. Architecturally inspired by CLIPS and can be used to access JavaBeans. Problem data stored as facts. "Reason" using IF…THEN…ELSE rules and can "reason" deductively (forward-chaining) or inductively (backward-chaining). JESS reasoner is free source for academic use only.

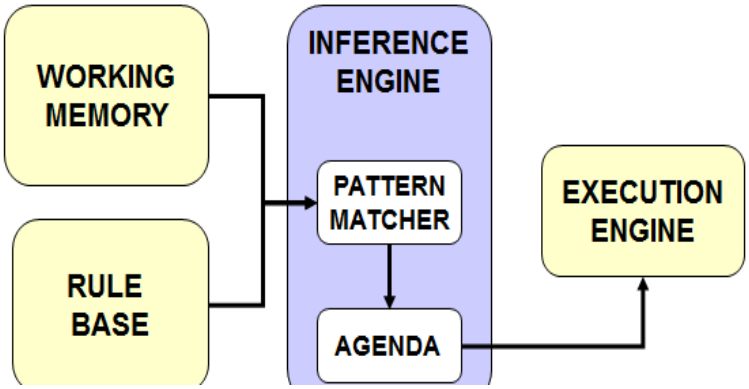


Figure 2.4 Jess Architecture Diagram (Abburu, 2012).

2.2 Introduction to Vehicular Ad hoc Network (VANET)

A Vehicular Ad hoc Network (VANET) is a wireless ad hoc network which provides communications among vehicles and nearby roadside equipment. VANET consists of vehicles with on-board sensors and roadside units (RSUs) deployed along highways/sidewalks, which provides communication between vehicle-to-vehicle (V2V) and communication between vehicles-to-infrastructure (V2I) (Kirthiga and Karthik, 2014).

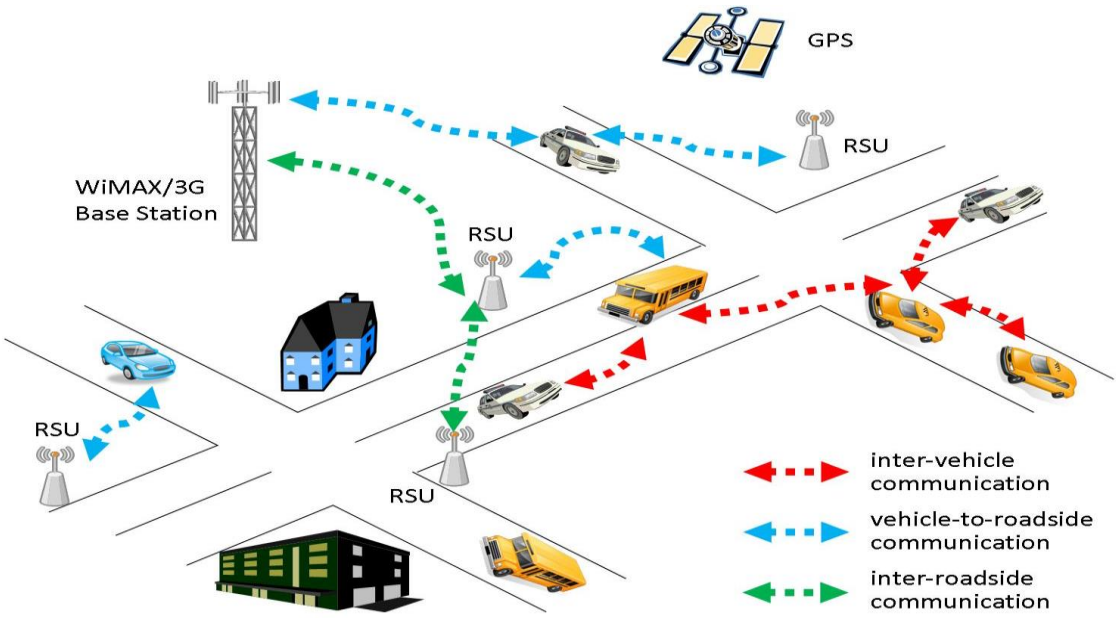


Figure 2.5 A Typical VANET Scenario (Kao, 2004)

Active Safety: send warning messages about dangerous traffic situations (such as an accident, road blocked, road congestion, sudden break and lane change).

2.2.1 Basic entities of the network

a. **Vehicle**: Vehicles communicate with each other for sharing local traffic information and improving the driving experience. A vehicle frequently requests the information from a Road Side Unit (RSU) or other vehicle and obtains enough information for the following period until passing by another RSU or other vehicles that can share information.

b. **On – Board Unit**: These are the vehicles having GPS, processing unit, radar, transmitting and receiving antenna.

c. **Roadside – Unit**: act as the infrastructure of VANET standing on the road side and connect with the trusted entity by wired links in the system.

2.2.2 Basic Communication Entities

a. Vehicle-to-Vehicle: V2V (vehicle to vehicle) is an automobile technology designed to allow automobiles to "talk" to each other. It is multi-hop multicast/broadcast communication used to transmit traffic related information over multiple hops to a group of receivers. ITS is generally concerned with the road ahead and not on the road behind. ITS mainly, uses two types of message forwarding techniques, Naive Broadcasting and Intelligent Broadcasting (Biswal, 2014; Xiaodong, 2007). V2V communication enable active safety systems that can assist drivers in preventing 76% of the crashes on the roadway, thereby reducing fatalities and injuries that occur yearly (Kirthiga and Karthik, 2014).

b. Naive Broadcasting: it considers periodic broadcasting of messages at regular intervals. If the message comes from behind then the vehicle ignores the message, but if the message comes from a vehicle ahead then the receiving vehicle send its own broadcast message to the vehicles behind it. The limitation of naive broadcasting method is that large numbers of broadcast messages are generated, hence increasing network overhead (Biswal, 2014).

c. Intelligent Broadcasting: overcomes the above limitation by using acknowledgements, hence limiting the number of message broadcasts. If the event-detecting vehicle receives the same message from behind it, it assumes that at least one vehicle in the back has received it and will be responsible for further transmitting the message, hence it ceases broadcasting (Biswal, 2014).

d. Vehicle-to-Infrastructure (V2I): The Road Side Unit (RSU) broadcasts message to the vehicles in the zone/area. Its kind configuration provides ample amount of bandwidth link between communicating parties. Frequently used for traffic optimization messages (Biswal, 2014; Xiaodong, 2007).

2.2.3 Types of Wireless Network

A network is a group of devices connected to one another. In the case of wireless networks, radio communication is frequently the medium of choice. Nevertheless, even within the radio-powered subset, there are many different technologies designed for use at different scales, topologies, and for different use cases. One way to illustrate this difference is to partition the use cases based on their "geographic range":

Table 2.1: Types of Wireless Networks

Type	**Range**	**Applications**	**Standards**
Personal area network (PAN)	Within reach of a person	Cable replacement for peripherals	Bluetooth, ZigBee, NFC
Local area network (LAN)	Within a building or campus	Wireless extension of wired network	IEEE 802.11 (WiFi)
Metropolitan area network (MAN)	Within a city	Wireless inter-network connectivity	IEEE 802.15 (WiMAX)
Wide area network (WAN)	Worldwide	Wireless network access	Cellular (UMTS, LTE, etc.)

Source: (Biswal, 2014)

2.2.4 Standards VANET communications

a. Dedicated Short Range Communication (DSRC): also called IEEE 802.11p, it is a developed standard by the IEEE and is a short to medium range communications service that is used for V2I and V2V communication.

b. Wireless Access in Vehicular Environment (WAVE): known as IEEE 1609.11p, in 2003, American Society for Testing and Materials (ASTM) sets ASTM-Dedicated Short Range Communication (DSRC) which was totally based on 802.11

MAC layer and IEEE 802.11a physical layer (Zhu and Roy, 2003). Vehicular scenario demands high speed data transfer and fast communication because of its high topological change and high mobility. It consists of Road Side Unit (RSU) and On-Board Unit (OBU). WAVE uses OFDM technique to split the signals.

c. Routing: It is a vast concept used in MANET and VANET environment. Many routing protocols have been designed for communication between the nodes in an ad-hoc environment. Routing protocols are divided into Topology Based, Position Based, Cluster Based, Geo Cast Based and Broadcast Based (Biswal, 2014; Xiaodong, 2007).

2.3 Related Works

There have been several studies on ontology based VANET. For example, in Saravanan *et al*., (2010), an ontology modelling approach for assisting vehicle drivers through safety warning messages during time critical situation was proposed. The work aimed at creating alert messages based on the context aware parameters using driving situation and such as driver's activity and environment.

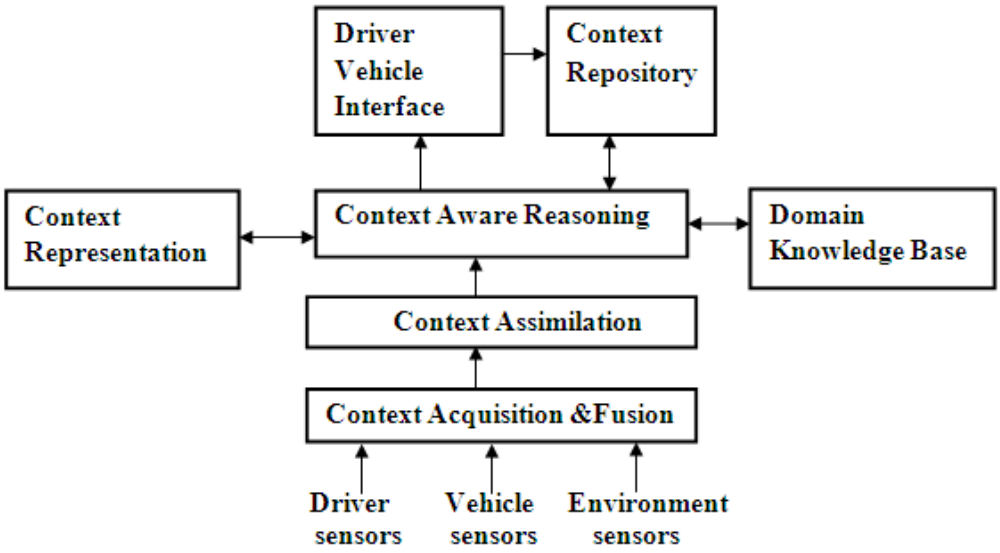


Figure 2.6 Architecture of Intelligent Driver Assistance System (Saravanan *et al*., 2010)

The work was implemented using ontology modelling and XML format that was used to manage the Intelligence- Driver Assistance System (I-DAS) parameter to illustrate the update /maintenance in the work. Java framework was used to generate safety alerts in several driving situations. The future work was to consider a more comprehensive extension of the overtaking knowledge base.

In the work of Morignot and Nashashibi, (2012) a high-level representation of automated vehicle, other vehicles and their environment was proposed which can assist drivers in taking such "illegal" (which means autonomous vehicle should follow its obstacle avoidance algorithm instead of traffic rule) but practical relaxation decisions. The high-level representation, ontology includes topological knowledge and inference rules, in order to compute the next high-level motion an automated vehicle should take as assistance to a driver. Where two instances were given, for example:

a. Situation where a vehicle was stuck on the road because of a vehicle ahead of it stopped on the lane with no good reason that might take a longer time (could be offloading things), meanwhile according to the traffic regulation the vehicle cannot overtake.
b. Circumstance on reaching a roundabout, a vehicle ahead of a driver stopped on the lane with an engine problem, and because of the traffic regulation the driver has to wait behind the defective vehicle until the car continues its movement again, which could take a longer hour to rectify.

Human drivers can manage these two situations (a) and (b) by checking the oncoming vehicle behind it, then reverse to the back and pull off the lane but in the case of autonomous vehicle driven by a computer, in these two instance (a) and (b) since the vehicle is considering the traffic regulation the vehicle will continue waiting (trapped) till the vehicle ahead it continues its movement again. In these situations, for an

autonomous vehicle to reason like a human driver (imitate) a decision needs to be taken, that is, should an autonomous vehicle follow its obstacle avoidance algorithm then change lane, or should it follow traffic regulation by staying on its lane?

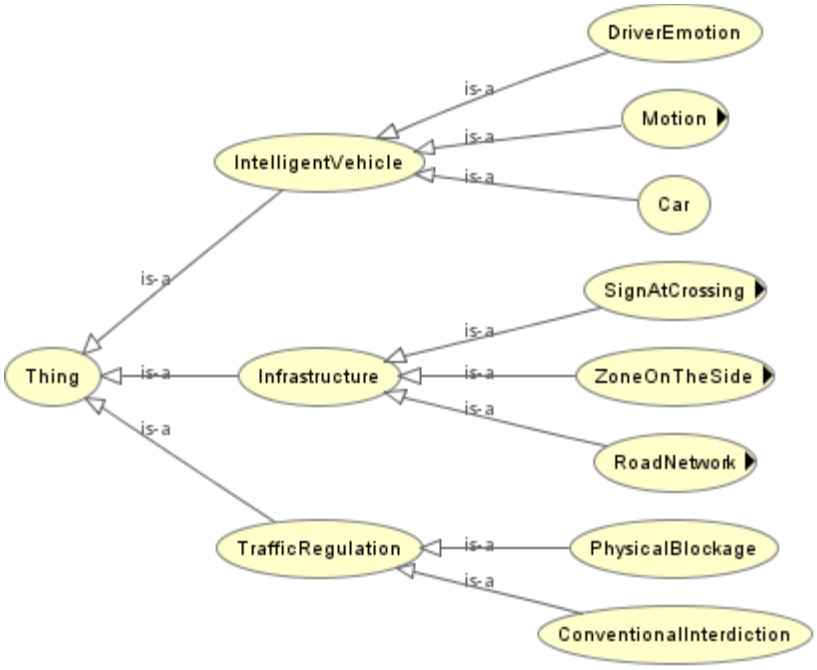


Figure 2.7 High-level Ontology Representation Model (Morignot and Nashashibi, 2012)

Table 2.2: Limitations Morignot and Nashashibi, (2012)

Technology used		Morignot and Nashashibi, (2012)
ADAS		Yes
ADAS&Human		No
Communication type design for the ontology	Leading vehicle	No
	SameDirection vehicle	No
	OppositeDirection vehicle	No
	RoadSideInfrastructure	No
	Pedestrian	Yes

Source: Morignot and Nashashibi, (2012)

The work focused on the internal part of each vehicle specifically its decisional part, it gives consideration for ADAS and pedestrian while the work do not considered given human driver advice, Leading vehicle, SameDirection, OppositeDirection and RoadSideInfrastructure communication.

Pollard *et al*., (2013) discussed about embedding a symbolic representation that is ontology as a component of each vehicle in order for it to deal with emergency situation and also to manage the situation at intersection in order to reason on the usage of traffic rules. It also discusses about the automation level of the vehicle such that it will observe the state of a driver from fully aware through drowsiness to full asleep by using a camera detecting eye opening level, blink frequency and blink duration. They later proposed to determine the maximum autonomy level that a vehicle can adopt in order to cope with the current state of the environment so that it can ensure a safe driving. Automation means all the modes which imply actions were done by the system through the actuators, including actions against the driver's intention such as emergency braking.

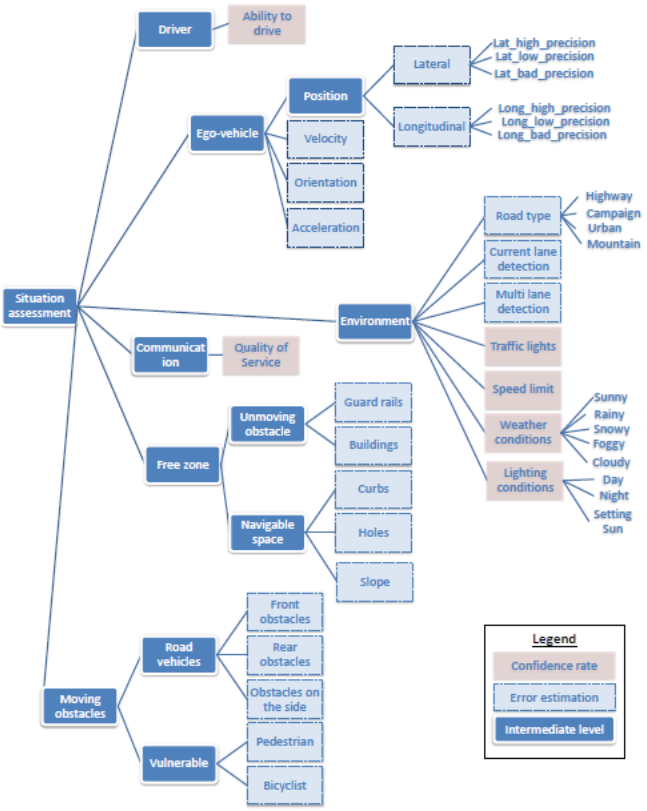


Figure 2.8 Ontology for Situation Assessment (Pollard *et al.*, 2013)

The limitations of this work were; it did not consider providing advice, warning to the driver, blind spot detection is not considered as an automation mode and communication between vehicles are not considered.

Ontology was used to model VANETS security in terms of identifying intrusion based on semantics (Erritali *et al.*, 2013). The work has shown different techniques of ontology-based intrusion detection, summary of ontology web language and how ontology can be used to create namespaces and declaration of class in order to identify the relationship between the parameter of vulnerabilities in term of intrusion detection.

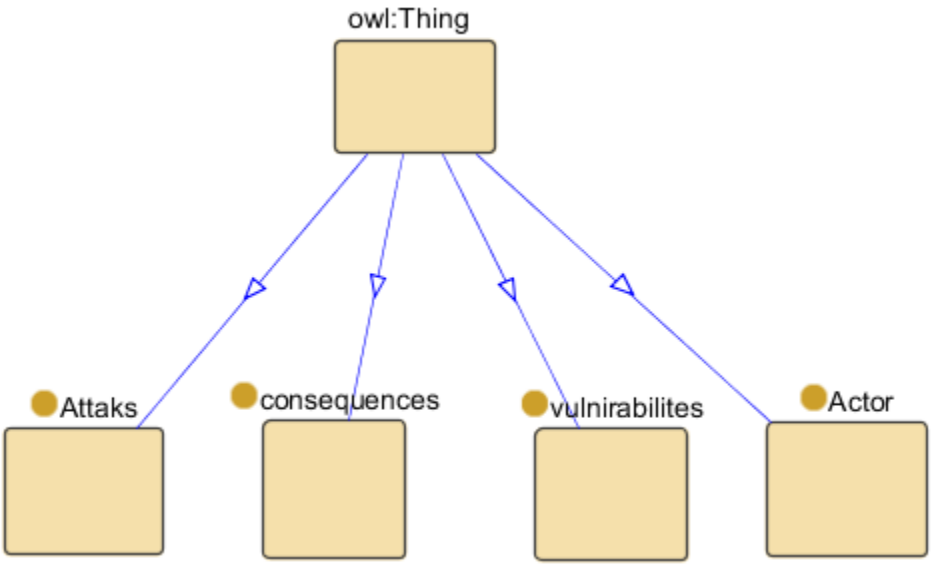


Figure 2.9 High Level Ontology (Erritali *et al*., 2013)

This work was based on use of ontology to solve intrusion detection for vehicular ad hoc networks. The work did not reflect on other aspect of ontology of VANET and their environment.

An ontology-based framework was proposed which provides human like reasoning about the driving environment from the facts gathered from on-board sensor, maps and vehicle state (Armand *et al*., 2014). This entails the information from different parameters available in road situations. The work uses terminological box (TBox) and an assertional box (ABox) to describe the concepts of their ontology in term of class and relationships between classes such as object properties and their rules. Also declare an instance of concepts such as individuals. The work used OWL to edit and verify the ontology in protégé and swoops. According to Armand *et al*., (2014) the objective was not to design an exhaustive ontology that considers every type of context entity, but rather to design a coherent, easily extendable ontology based framework and it can only reason on contexts compatible with it, that is contexts which only meet entities which

have been defined in the TBox. This means that for an intensive use of the ontology, it has to be extended to take new types of entities into consideration.

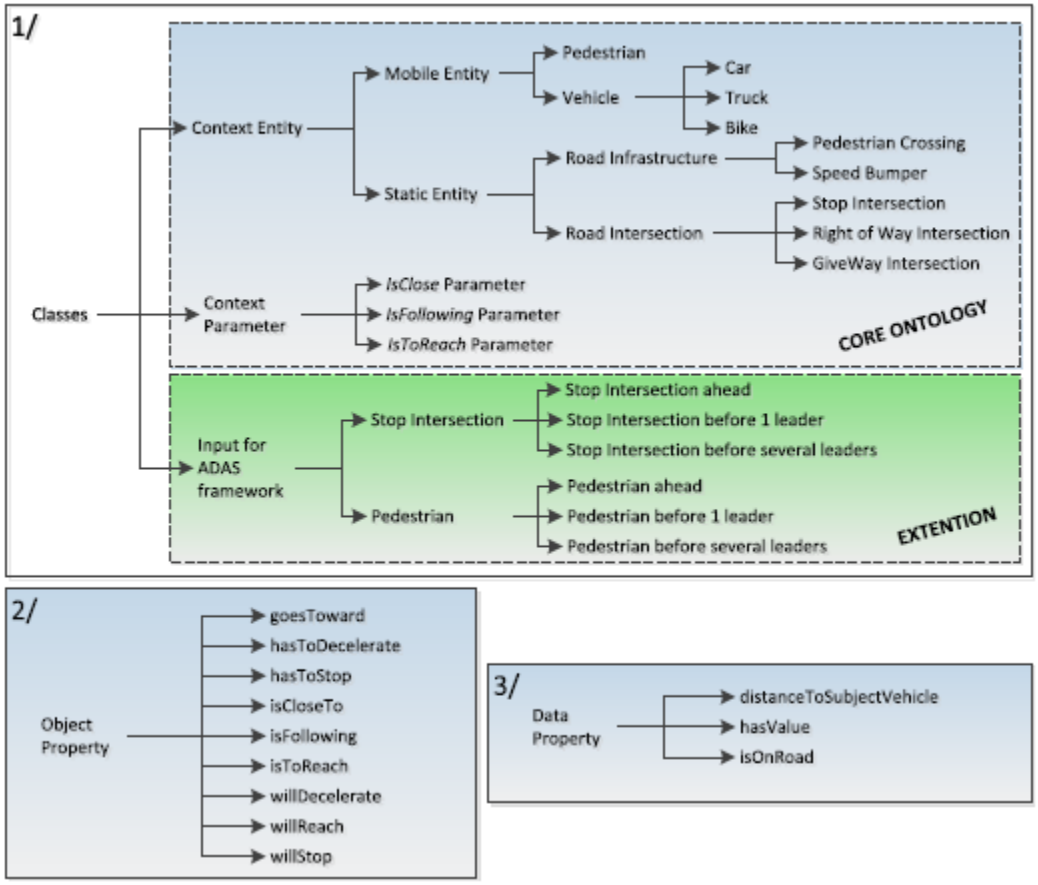


Figure 2.10 Ontology Classes, Object and Data Properties (Armand *et al*., 2014)

Table 2.3: Limitations Armand et al. (2014)

Technology used		Armand *et al. (*2014)
ADAS		Yes
ADAS&Human		No
Communication type design for the ontology	Leading vehicle	Yes
	SameDirection vehicle	Yes
	OppositeDirection vehicle	No
	RoadSideInfrastructure	Yes
	Pedestrian	Yes

Source: Armand *et al. (*2014)

The work primarily based on ADAS (Advance driving Assistance System) was concerning the interaction between the lead vehicle, the pedestrian and the road intersection. The aim of this dissertation to improve performance in VANET system through the development and implementation of an enriched VANET ontology knowledge base which will extend the issue of lead vehicle considering ADAS, vehicle at the opposite lane and as well as gives a suggestion on road usage for a human driven vehicles.

2.4 Existing System

Uschold and Gruninger, (1996), Noy and McGuinness, (2001) and Brusa *et al*., (2006), discuss several research on existing methods of developing ontology, such as EO (Ontology Enterprise), TOVE (TOronto Virtual Enterprise), KACTUS (also called CML (Conceptual Modeling Language)) and METHONTOLOGY.

The research results noted that there is no one correct way to model a domain that is, there are variety of different approaches of developing ontology which makes researcher to use methodology based on the application in their mind.

Figure 2.11 shows the development process adopted by many researchers.

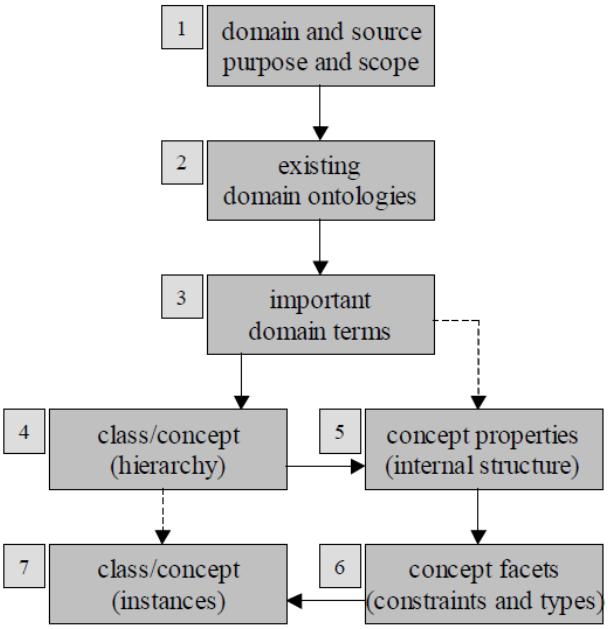


Figure 2.11 Ontology Development Process (Boyce and Pahl, 2007)

CHAPTER THREE

ONTOLOGY DEVELOPMENT AND RULE SYSTEM FOR VANET

3.1 Introduction

This chapter discusses the methodology for designing the proposed communication channel or protocol and VANET ontology that will be used alongside with the rule. The architecture of the system upon which the rule set and the ontology being designed were to be embedded, was also presented in this chapter. Subsequent sections further outlines the test bed questions upon which the rules or the communication protocol were built upon. A closer look is also presented on the how SWRL and OWL were interpolated together in realizing the objectives of this dissertation.

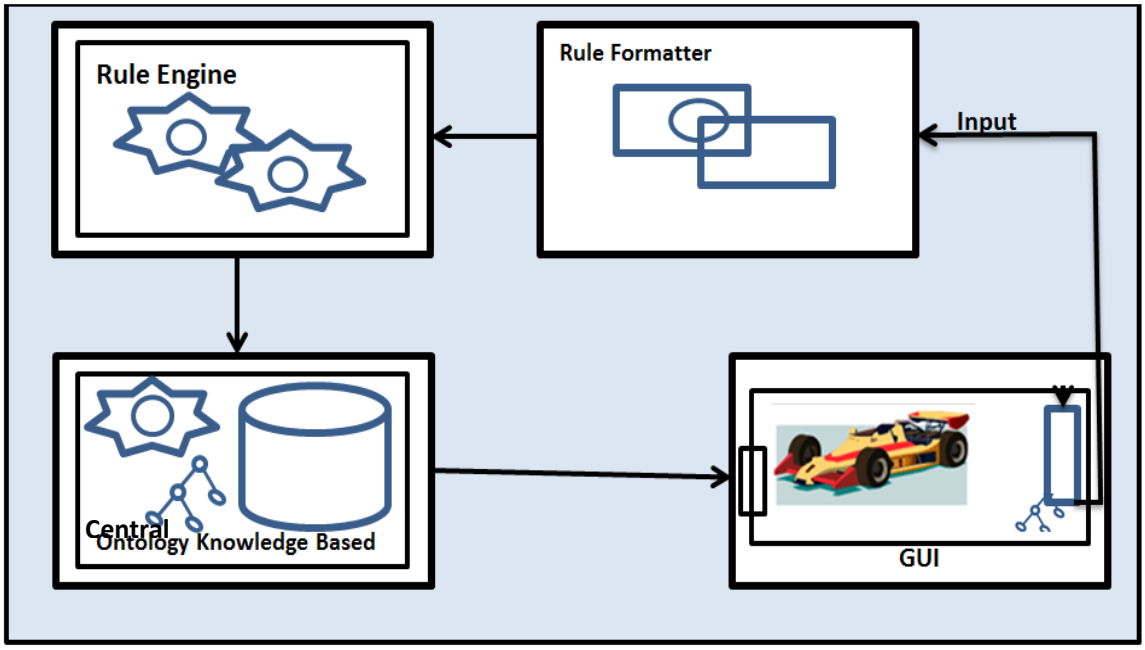


Figure 3.1 System Architecture

3.2 System Architecture of Semantic Web Based VANET

This section gives the detail of the architecture of the proposed system. The architecture is divided into four main components, namely:

a. **GUI (Graphic User Interface)**: The GUI (from Figure 3.1) of the proposed system presents a prototype of what might have been seen as a simulated autonomous vehicle environment. For the purpose of demonstrating the communication pattern that could exist between vehicles in a VANET system, the GUI is provided as means of invoking some possible events and actions that generate communication data. The component of the overall system is compartmentalized into two views: the control and event output panels. The control panel provides users with some control keys. Each of this key is denoting a possible action or event that could be thrown in a normal driving scenario. Section 3.5.2 lists out the test bed questions that most of these control keys stand to represent. The second panel details the output of the control key sets that must have being triggered. For instance, should a user trigger a control key denoting change of lane, the ontology must first be updated that vehicle A has charged lane. This ontology update is done through the rule set or communication protocol proposed in this dissertation. Once this is noted in the ontology, other vehicles in the VANET system will be alerted so as to maintain safety precautions within the system. Meanwhile, in the event output panel, which in itself a canvas for drawing object, an action and reaction effects are seen based on the move of this control key. The overall emphasis of this synergy is to demonstrate that VANET ontology could be employed beyond being just a knowledge base to being a communication channel where all vehicles participating in the system update themselves on the state of the system. Other possible events being abstracted in the control key panel are;

i. If it is a roundabout, then the vehicle needs to slowdown

ii. If it is a pedestrian or zebra crossing, the vehicle needs to stop

iii. If it is line crossing, the vehicle needs to stop. Which means the speed at which the car is moving will decrease, depending on the situation that will soon occur.

The ontology shown within the GUI component of the architecture is to imply that each vehicle in the VANET system has its own miniaturized ontology. This ontology models its personal details and information retrieved from a centralized ontology. The purpose for a centralized ontology is for systemic information sharing. While the aim of the miniaturized ontology for each vehicle is to decentralize each node's (vehicle) self-tuned data – this engenders autonomous-city of each vehicle in a data sharing system.

b. Inputs and Rule Formatter: Data entry are made through the control keys, however, each action triggered by these keys must be semantically interpreted as queries which are in turn formatted in rule format. These formatted rules are then run against the underlying ontology so as to make a fact known. This component in the system architecture illustrated by Figure 3.1 mediates between the knowledge base and the nodes (vehicles) in the VANET system. Hence the communication protocol proposed in this dissertation.

c. Rule Engine: In Semantic web, rules are implemented against ontology by a rule engine. There are quite a number of rule engines and reasoners, and these include Pellet, Jess, Jena, Helmitt and FACT++. However, this dissertation employs the use of Jess as a rule engine because of its support for SWRL. Rule engine, also called a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. It provides automated support for reasoning tasks such as classification, debugging and querying. In Figure 3.1 rule engine inferred logical inference on rule created in section 3.5.2 against the ontology designed (Figure 3.8) to reason out information.

d. Central Ontology Knowledge Base: Most possible concepts or objects in the domain of VANET are being modeled in this ontology. This ontology includes both roadside objects and basic elements of a typical driving environment. All relating data are stored in this ontology. Inter-vehicular communication is attained through information sharing provided through this model. Instances of the classes modeled are added through every input made by each node or vehicle in the system. The observation is that both vehicles and some signal emitting roadside objects constitute what is being identified as nodes in this system. This is a special type of database that holds information representing the expertise of the ontology domain developer. All the information are been reported to the central ontology that is, any passing car can report information based on what it has encountered from the direction where it is coming from or from the source where it is coming from, so that other cars, or vehicles can read from this central ontology which is installed on the roadside unit (infrastructures).

3.3 Building a Knowledge Base for VANET

This section illustrates how ontology development methodology is being done, it gives the details of the VANETs ontology concepts and also explains VANETs domain knowledge and how it is being built.

3.3.1 Ontology Development

Ontology can further be defined as a formal explicit description of concept in a domain of discourse (class), properties of each concept describing various features and attributes of the concept (slot), and restrictions on slots (facets). Ontology together with a set of individual instances of classes constitutes a knowledge base. Classes are the focus of

most ontology. Classes describe concepts in the domain and slots describe properties of classes and instance. As suggested in the research work of Noy and McGuinness, (2001) and Boyce and Pahl, (2007) in figure 2.11 the following steps are relevant in the process of developing as well as designing ontology:

a. Determine the domain and source, purpose and scope of the ontology.
b. Consider the reuse of existing ontologies.
c. Enumerate important terms in the ontology.
d. Define the class / concept hierarchy.
e. Define the properties of the classes (Internal structure).
f. Define the concept facets of the properties (constraints and types).
g. Create class / concept (instances).

This will be used in developing the ontology of this dissertation in OWL application with protégé editor.

3.3.2 VANETs Ontology Concepts Enumeration

a. **Communication** as one of class hierarchies of the ontology of VANETs has Warning and RoadBlocked as subclasses, while Warning has Accident as subclass and RoadBlocked has Congestion as it owns subclass in the ontology.

b. **Direction** as one of the class hierarchies has OppositeDirection and SameDirection as its subclasses in the ontology.

c. **Place** as one of the class hierarchies of the ontology has Destination and Source as it subclasses in the ontology.

d. **RoadSide Infrastructure** is a class of the ontology of VANETs without any subclass.

e. **Vehicle** has PrivateVehicle and PublicVehicle as its subclasses in the ontology.

f. **Roadobject** as one of the class hierarchies of the ontology has Lane, Roundabout and TrafficControlSystem as it subclasses, while Lane has Lane1 and Lane2 as its subclasses and TrafficControlSystem has no subclass.

g. **Alert** as GreenLight, Off, RedLight and YellowLight as its subclasses in the ontology.

h. **MotionState** is a class of the ontology of VANETs without any subclass.

3.3.3 VANET Domain Knowledge

The first step is to determine the domain and source, purpose and scope of the ontology to be designed. And for this ontology, the domain is Vehicle information and the scope is to model the basic and necessary information about road information. This information includes the direction, place, roadside infrastructure, vehicle, and roadobject.

The second step requires the consideration of reusing existing ontology if there is. However, there is no access to any existing ontology that this research could exploit and use.

The third step is to enumerate the possible terms and entities. It has been enumerated in section 3.3.2. Figure 3.2 demonstrates the classes in their hierarchy as demanded by step four in ontology design and Figure 3.4 shows how classes are created.

Step five shows how relations (properties) are created to connect entities or classes and literal values. Figure 3.3 illustrates how to create properties.

There are two types of properties (relations) that are used in OWL. These are DataProperty and ObjectProperty. Some of the properties used here are isopposite,

communicates, moves, movingtowards, plies, signalsvehicle, takingroundabout and warnsvehicle.

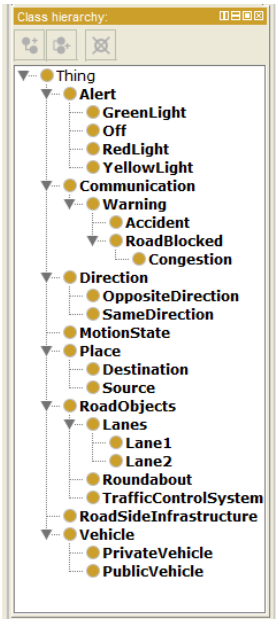


Figure 3.2 Class Hierarchy

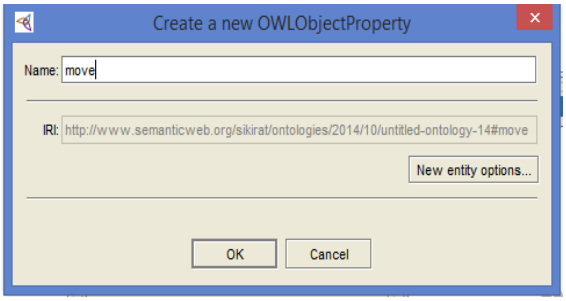


Figure 3.3 Creating ObjectProperty

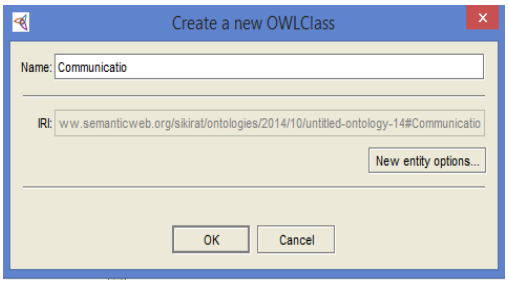


Figure 3.4 Class Creation

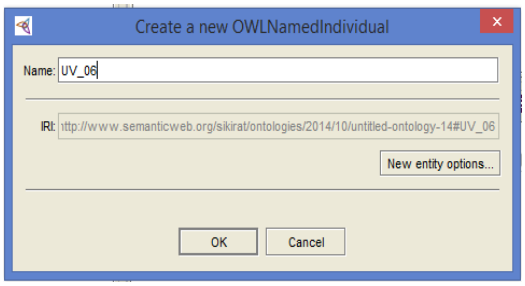


Figure 3.5 Adding Instances to Ontology

Figure 3.5 illustrates how individual were added to these entities or classes created. Figure 3.6 shows object property created. Figure 3.7 shows the individual Member's lists for one of the classes.

The last step creates the individuals or instances of the classes. Figure 3.8 shows the ontology design, as depicted with Graphviz.

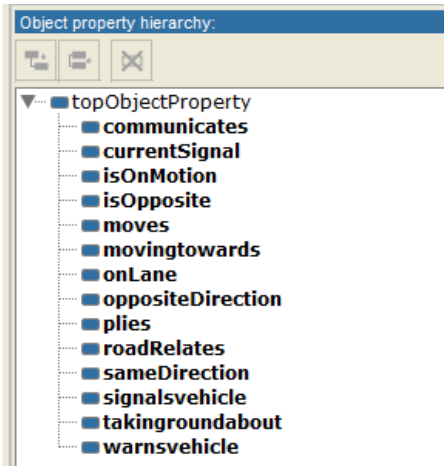


Figure 3.6 Object Property

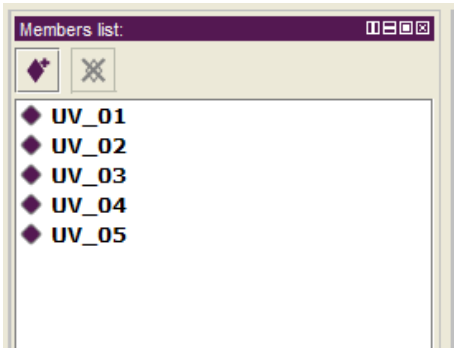


Figure 3.7 Individual Members List

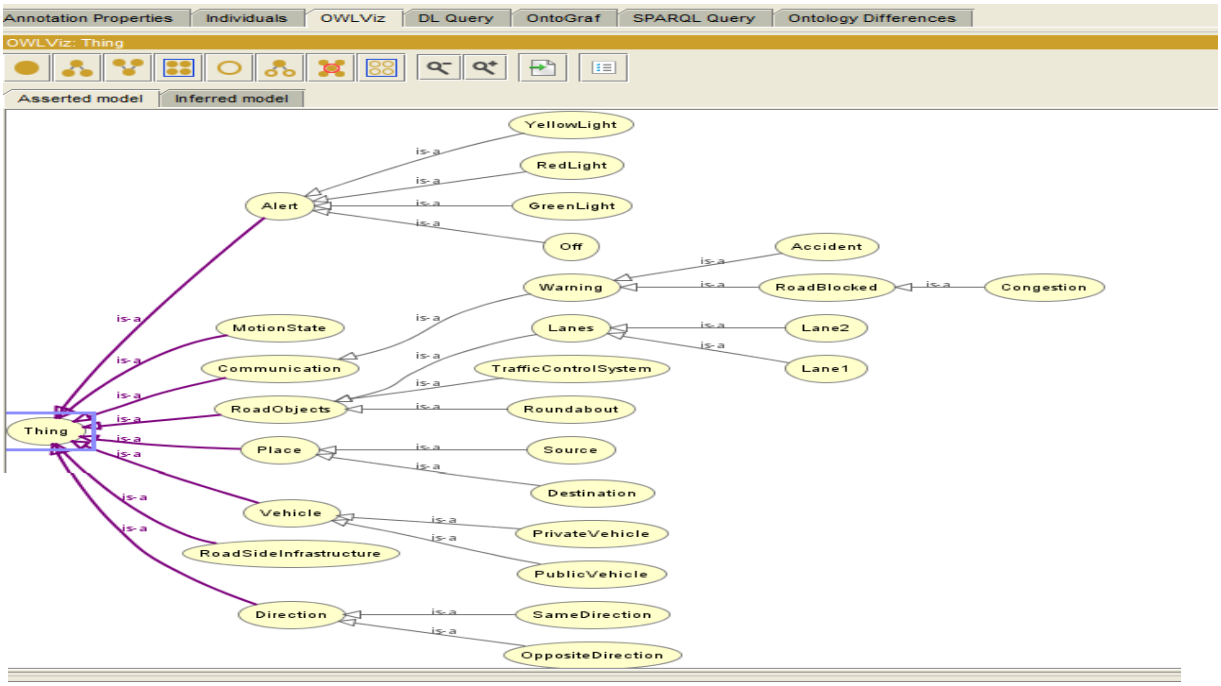


Figure 3.8 The Ontology Designed

3.4 A Protocol for Dynamically Populating VANET Ontology

Protocol-V: Insert(VehicleOntology: VehicleClass, CentralOntology: VehicleClass)
VehicleParameterInstance (VehicleName, Direction, StartingLane, Source, Destination, Distance).

Protocol-R: Insert(CentralOntology: RoadObjectClass)
RoadObjectParameterInstance (Lane1, Lane2, Roundabout, TrafficControlSystem)

Protocol-D: Insert(CentralOntology:DirectionClass, VehicleOntology:DirectionClass)
DirectionParameterInstance (OppositeDirection, SameDirection)

Protocol-C: Insert(CentralOntology:CommunicationClass)
CommunicationParameterInstance (Accident, Congestion)

Protocol-I: Insert(CentralOntology:RoadSideInfrastructureOntology)
RoadSideInfrastructureParameterInstance (nil)

Protocol-P:Insert(CentralOntology:DisplacementClass,
VehicleOntology:DisplacementClass)
PlaceParameterInstance (Source, Destination)

Protocol-P:Insert(CentralOntology:AlertClass, VehicleOntology:AlertClass)
AlertParameterInstance (GreenLight, Off, RedLight, YellowLight)

Protocol-P:Insert(CentralOntology:MotionStateClass,
VehicleOntology:MotionStateClass)
MotionStateParameterInstance (Nil)

The composition of these protocols is to theoretically demonstrate how instances were semantically added into the corresponding ontology they relate with. The syntax of the protocol usually starts with the *Insert* keyword. This then follows with parenthesis and a

list of the ontology files, with their classes, that are going to receive the supplied instances. An ontology name is separated from its class, object or concept by the : notation. While a comma separates a list of ontology files to be updated. The instances that were added to the selected class in the ontology were likewise listed in parenthesis, separated by comma.

3.5 Event Control in VANET through Rule System

In this section basic interoperability of SWRL and OWL, the proposed VANET rule set created and functionality question and definition were discussed.

3.5.1 Basic interoperability of SWRL and OWL

There is limitation of natural ability of expression for the ontology and basic rule SWRL which makes them work hand in hand such that the rules are good at pluralistic relations with deductive knowledge, and ontology plays a role on the description of the unary concept since the rules do not support the concept of existence, ontology therefore, provides a complex definition of this concept of existence.

The SWRL Editor itself does not perform any inference. However, a bridge mechanism is provided to allow interoperation with rule engines. The Jess rule engine is supported, and is accessible through the SWRLJessTab.

The ontology in this dissertation is designed with protégé 3.4.8 and 5.0 editors which were designed with OWL language.

3.5.2 Proposed VANET rule set

Table 3.1 VANETs System Functionality Question

A	Given a particular vehicle and a distance apart, get the data of vehicles which are both on the same lane and parallel lane.
B	Which roadside objects are close to a given moving vehicle?
C	Is the vehicle on motion and in what direction?
D	Which data has vehicle moving in opposite direction: Lane 1 Lane 2
E	Is vehicle at crossroads? Then what is the data from traffic control system.
F	What is the average speed to move with in respect of roadside objects?

In this dissertation, SWRL is used to create rules and SQWRL is used to support OWL queries. Now, a close look is given to rule formulation and how they can be combined with the query language SQWRL in retrieving statements from the underlying ontology. The rule takes the format of the popular Horn like structure which comprises of both the antecedent and consequent. The antecedent is on the left hand side while the consequent is placed on the right hand side. Provided that a truth value or result is obtained from the antecedent, then the consequent follows, otherwise, no execution of the consequent.

Here, the outlines of the rules are shown for further implementation in the next chapter.

A. Given a distance apart, get the information of vehicles which are both same lane and parallel lane.

i. Vehicle(?v) ^ onLane(?v, Lane2) ^ sameDirection(currentVehicle, ?v) ^ distance(?v, ?d) ^ swrlb:lessThanOrEqual(?d, 5) → sqwrl:select(?v)

ii. Vehicle (?v) ^ onLane2 (?v, Lane2) → sqwrl: select (?v)

iii. Vehicle (?v) ^ onLane1 (?v, Lane1) → sqwrl: select (?v)

B. Which roadside objects are close to a given moving vehicle?

i. RoadObjects(?obj) ^ roadRelates(?obj, currentVehicle) ^ distance(?d, ?obj) ^ swrlb:lessThanOrEqual(?d, 10) → sqwrl:select(?obj)

ii. RoadObjects(Roundabout) ^ roundrunrate(Roundabout, ?kmh) → speedrate (?km, currentVehicle)

iii. RoadObjects(Pedestrian) ^ pedestrianrunrate(Pedestrian, ?kmh) → speedrate (?kmh, currentVehicle)

C. Is the vehicle on motion, and in what direction?

i. Vehicle(?v) ^ isOnMotion(?v, Motion) ^ plies(?v, ?lane) → sqwrl:select(?lane)

D. To obtain information of a vehicle moving in an opposite direction:

i. Vehicle(?v) ^ oppositeDirection(currentVehicle, ?v) → sqwrl:select(?v)

The direction and lane plied by this vehicle will be programmatically decoded. This will be shown in chapter four.

E. What is the average action to take with respect to traffic control system signal?

i. RoadObjects(?obj) ^ abox:hasClass(TrafficControlSystem, ?obj) ^ currentSignal(?obj, ?signal) ^ averageSpeed(?signal, ?r) → sqwrl:select (?r, ?signal)

F. What is the average speed to move with in respect of destination or roadside objects?

i. RoadObjects (?r) ^ averageSpeed (?r, ?speed) → sqwrl:select (?r, ?speed)

ii. Vehicle(?v) ^ pathRelates(?v, currentVehicle) ^ distance(?v, ?d) ^ swrlb:lessThanOrEqual(?d, 5) → sqwrl:select (?v)

The outlined rules which are first of all transformed from plain text to SWRL statement, then the rule engine (JESS) inferred logical inference on the rule against the ontology designed in figure 3.8 to provide reasoning of information which may not be included the designed ontology (such as if a vehicle is intending to follow lane A, then reverse through lane B and eventually follows lane A, the rule engine have to reason it out).

CHAPTER FOUR

IMPLEMENTATION AND DISCUSSION

4.1 Introduction

This chapter gives the details of the implementation of the design, according to the proposed system, which discusses the coupling of the rule engine with the development environment, how the semantic web based VANET was done, semantic web based VANET application was applied and also the final discussion and results.

4.2 Implementation Tools: Protégé, NetBeans and Jess

The tools used in carrying out the implementation of the theoretical framework discussed in the chapter two are hereby enumerated. Protégé is used in developing the ontology upon which the rules and queries sources their information. NetBeans IDE is a free and open source integrated development environment which has proven to be efficient in project development and well accepted by many programmers or developers because of its flexibility and its provision of built-in support. To implement the application in this dissertation, NetBeans 8.0.2 was deployed. Jess is the rule engine used for implementing the rules and queries outlined in chapter three. The next section shows the necessary steps carried out in coupling the Jess rule engine with the implementation of the simulated VANET environment developed in this research.

4.3 Coupling of Rule Engine with Development Environment (NetBeans)

Figure 4.1 shows the package listing of the Java files that were written in developing the simulated VANET environment. To configure our rule engine (Jess) with the source code, the jar files that consist of the rule engine are added to the package/project to which the implementation is being executed. Figure 4.2 shows how to navigate to the NetBeans section for adding libraries to a project. To add the rule engine jar files, the Add Jar file button is clicked, and then all the jar files are added by selecting them, and then clicking on the Open button. Once this jar files are added, the source codes in the project automatically import the necessary packages/API they have being assigned to use.

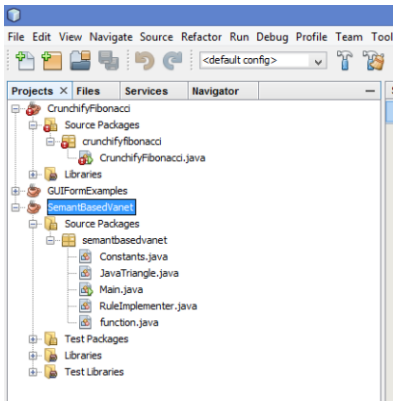


Figure 4.1 Package Listing of the Java Files

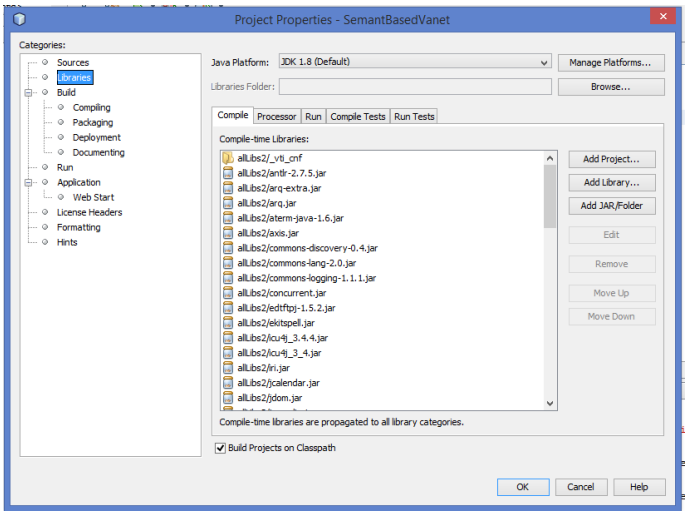


Figure 4.2 Navigate the NetBeans Section for Adding Libraries to a Project

Figure 4.3 shows a listing of the Jess rule engine jar files added to the project that implements the simulated VANET environment. While Figure 4.4 shows how some of the classes and packages of the Jess rule engine are being imported into the Java files written for this research.

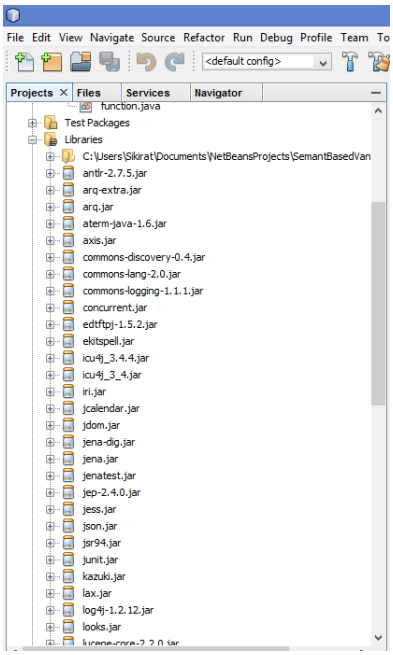


Figure 4.3 Listing of the Jess Rule Engine jar Files

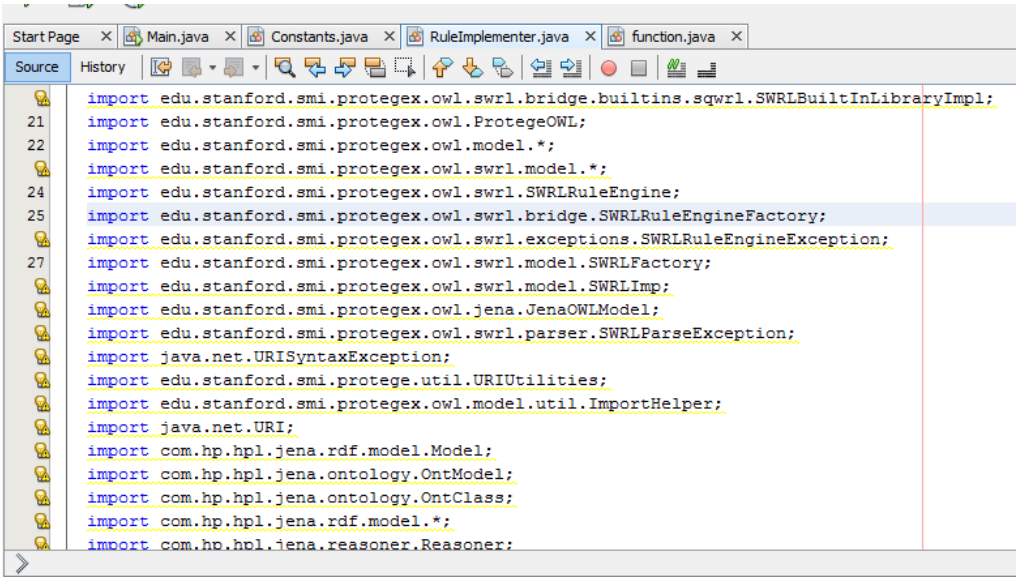


Figure 4.4 Classes and Packages of the Jess Rule Engine Imported into the Java File

4.4 Implementation of the Semantic Web Based VANET

Figure 4.5 shows the semantic rule set developed for the VANET system used for the ontology of the knowledge repository, which was created using SWRL and queried with SQWRL language.

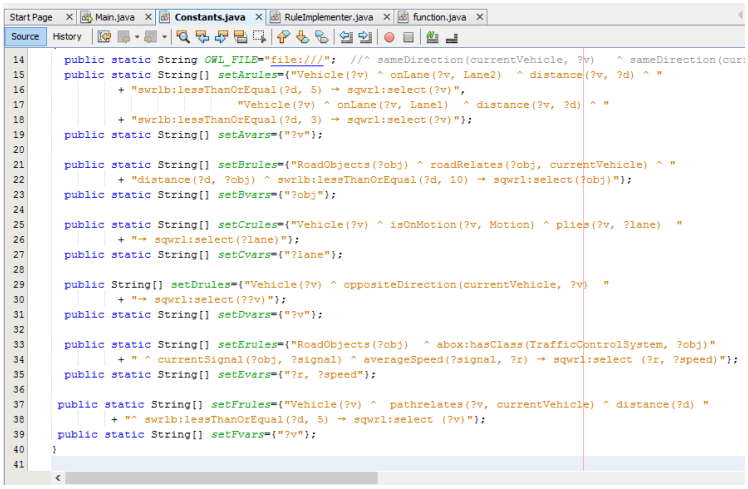


Figure 4.5 Semantic Rule Set for the VANET System

4.4.1. VANET communication channel with ontology

Figure 4.6 shows the vehicle ontology file developed with protégé opened in Java Netbeans development environment.

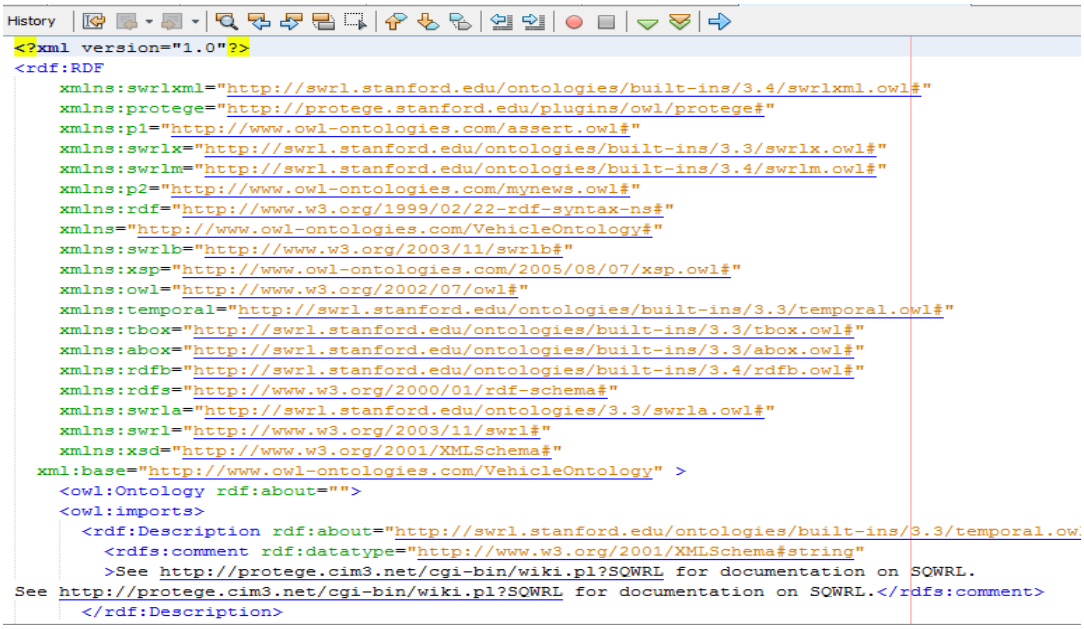


Figure 4.6 Vehicle Ontology File Opened in Netbeans Environment

4.4.2 Rule set formatting process

Figure 4.7 shows the rule formatting in development environment, Figure 4.8 and Figure 4.9 shows the main source code for developing the GUI in Java programming language to implement the VANET system. It takes care of the connection of semantic rule set, it queries and the process of adding vehicle was implemented in the program.

```
            String[] setArules={"Vehicle(?v) ^ onLane(?v, Lane2) ^ sameDirection("+currentVehicle+", ?v) ^ distance(?v, ?d) ^
swrlb:lessThanOrEqual(?d, 5) -> sqwrl:select(?v)"};
                          //,"Vehicle(?v) ^ onLane(?v, Lane1) ^ sameDirection(currentVehicle, ?v) ^ distance(?v, ?d) ^ swrlb:lessThanOrEqual(?d,
3) -> sqwrl:select(?v)"};
                String[] setAvars={"?v"};

               String[] setBrules={"RoadObjects(?obj) ^ roadRelates(?obj, "+currentVehicle+") ^ distance(?d, ?obj) ^ swrlb:lessThanOrEqual(?d, 10
-> sqwrl:select(?obj)"};
                String[] setBvars={"?obj"};

                String[] setCrules={"Vehicle(?v) ^ isOnMotion(?v, Motion) ^ plies(?v, ?lane)  ? sqwrl:select(?lane)"};
                String[] setCvars={"?lane"};

                String[] setDrules={"Vehicle(?v) ^ oppositeDirection("+currentVehicle+", ?v)  ? sqwrl:select(??v)"};
                String[] setDvars={"?v"};

                 String[] setErules={"RoadObjects(?obj)  ^ abox:hasClass(TrafficControlSystem, ?obj) ^ currentSignal(?obj, ?signal) ^ averageSpeed
(?signal, ?r) ? sqwrl:select (?r, ?speed)"};
                 String[] setEvars={"?r, ?speed"};

                 String[] setFrules={"Vehicle(?v) ^  pathrelates(?v, "+currentVehicle+") ^ distance(?d) ^ swrlb:lessThanOrEqual(?d, 5) ?
sqwrl:select (?v)"};
                  String[] setFvars={"?v"};
```

Figure 4.7 Rule Set in Development Environment

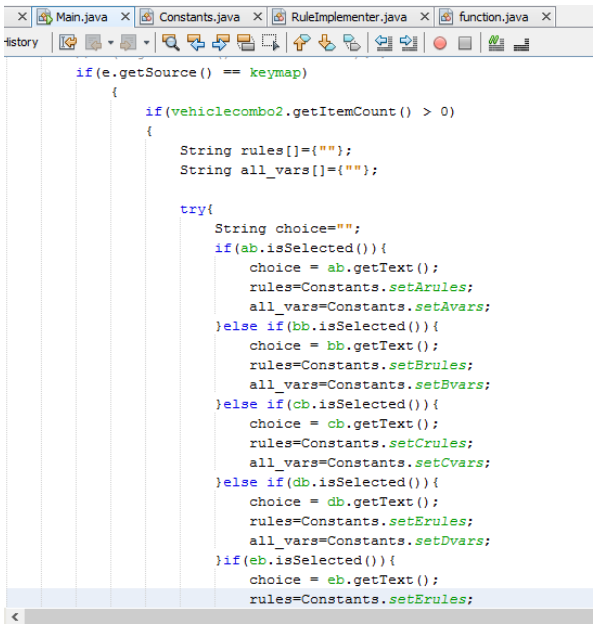


Figure 4.8 Development of the GUI in Java

```
61    JPanel side1 = new JPanel();
62    JPanel side2 = new JPanel();
63    JTextArea notepad=new JTextArea();
64    JTextArea output_cmd=new JTextArea();
65    JButton loadont = new JButton("Load Ontology");
66    JButton close = new JButton("Close");
67    JButton simulate = new JButton("Simulate");
68    JButton keymap = new JButton("Rule Map");
69    JButton addvehicle = new JButton("Add Vehicle");
70    JButton keymapb = new JButton("Rule Map");
71    JLabel vehiclelabel = new JLabel("Vehicles");
72    JLabel canvaslabel = new JLabel("Canvas");
73    JLabel vehiclecombo2lbl = new JLabel("Apply rule to:");
74    JLabel blocklabel = new JLabel("CMD or Black Background \t \t");
75    String veh[];//={"Choose One","VO 1","VO 2","VO 3","VO 4"};
76    JComboBox vehiclecombo;// = new JComboBox(veh);
77    static String currentVehicle;
```

Figure 4.9 Development of the GUI in Java

4.4.3 Rule set implementation

Figure 4.10 shows the source code in Java to add rule set in connection with the development environment.

```
public String queryWitSQWRL(String qry, String[] vars, String qry_name)
{
  OWLModel ontModel=getOWLFile();
  String[] all_vars=vars;

  StringBuffer output=new StringBuffer();
  try {
        SQWRLQueryEngine queryEngine =SQWRLQueryEngineFactory.create(ontModel);
        SWRLFactory fact = new SWRLFactory(ontModel);
        SWRLRuleEngine ruleEngine = SWRLRuleEngineFactory.create(ontModel);
        queryEngine.createSQWRLQuery(qry_name, qry);
        SQWRLResult result = queryEngine.runSQWRLQuery(qry_name);
        if(!result.hasNext())
        {
          while (result.hasNext())
            {
              for(int n=0; n < all_vars.length; n++)
                 output.append("\t\n"+((IndividualValue)
                    (result.getObjectValue(all_vars[n]))).toString());

              result.next();
            }
        }
     }
catch(Exception ex) { System.out.println(ex); }
```

Figure 4.10 Rule Set Query Implementation in Java

4.5 The Semantic Web Based VANET Application

This section discusses about implementation of the two fold panels, the control panels key activation and their actions and reaction of key activation in the GUI of semantic web based VANET application system.

4.5.1 The two – fold panels

Figure 4.11 shows the two – fold panels: The control keys panels and event output view panels. First of all before the control key can be activated, ontology file has to be loaded by clicking on load ontology button then choose the ontology file on the desktop or from the source it is located, the white board under the control keys will show the ontology file loaded. The control panel provides user with some control keys. Each is denoted with its possible action such as the key vehicles which have many options that user can choose from PublicVehicle_01 to PublicVehicle_05 and PrivateVehicle_01 to PrivateVehicle_05, then user can click on add vehicle button to insert the vehicle. There is also an apply rule section where user can select any rule (rule discussed in section 3.5) from A to F from the rule set for the ontology developed then click on rule map to apply the rule. The event view panels show the output of vehicles added in the two lanes while the black background under the output panel shows the output of the ontology update from the rule maps activated. The closed button is used to close the GUI.

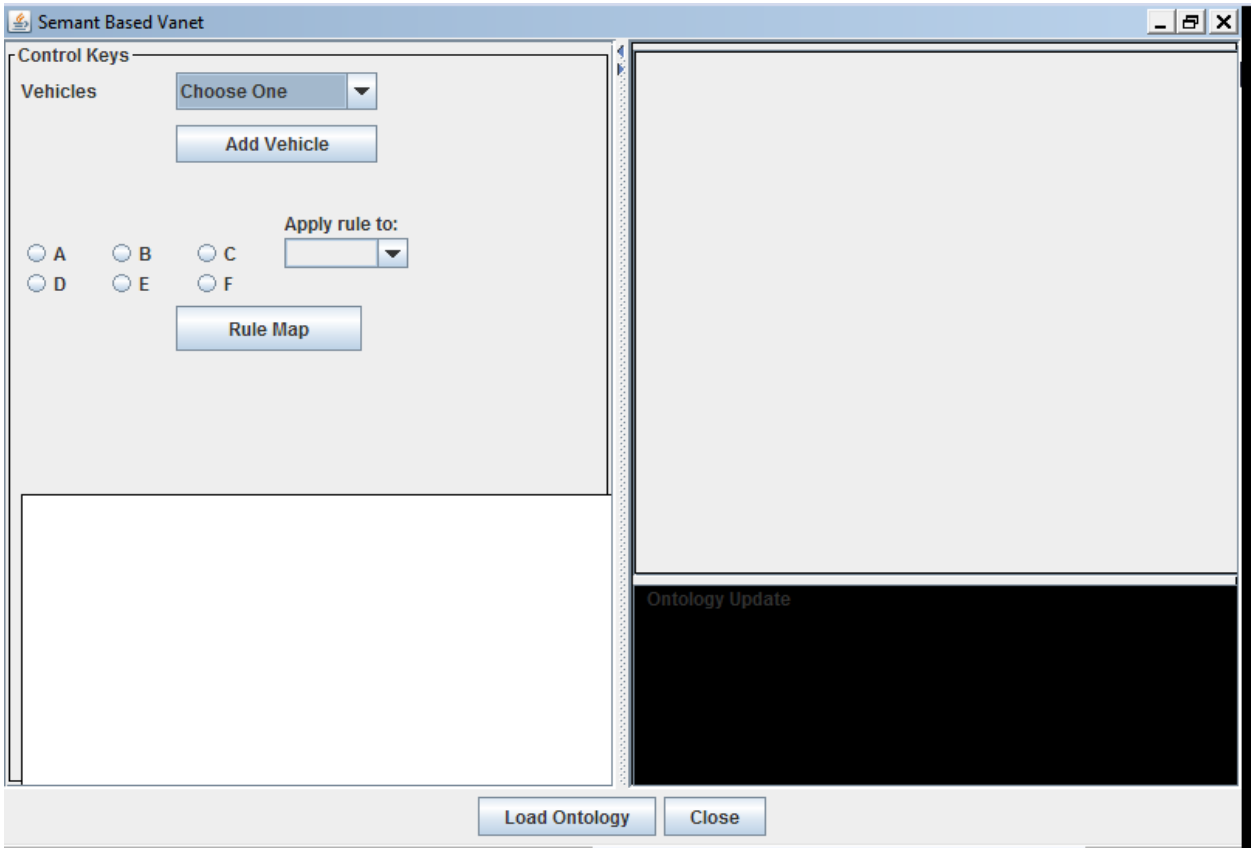


Figure 4.11 Two – Fold Panels GUI

4.5.2 Control panels key activation

Figure 4.12 through Figure 4.23 shows the outcome of control key activation for ontology load view, output view for vehicle and ontology update view GUI as discussed in section 4.5.1.

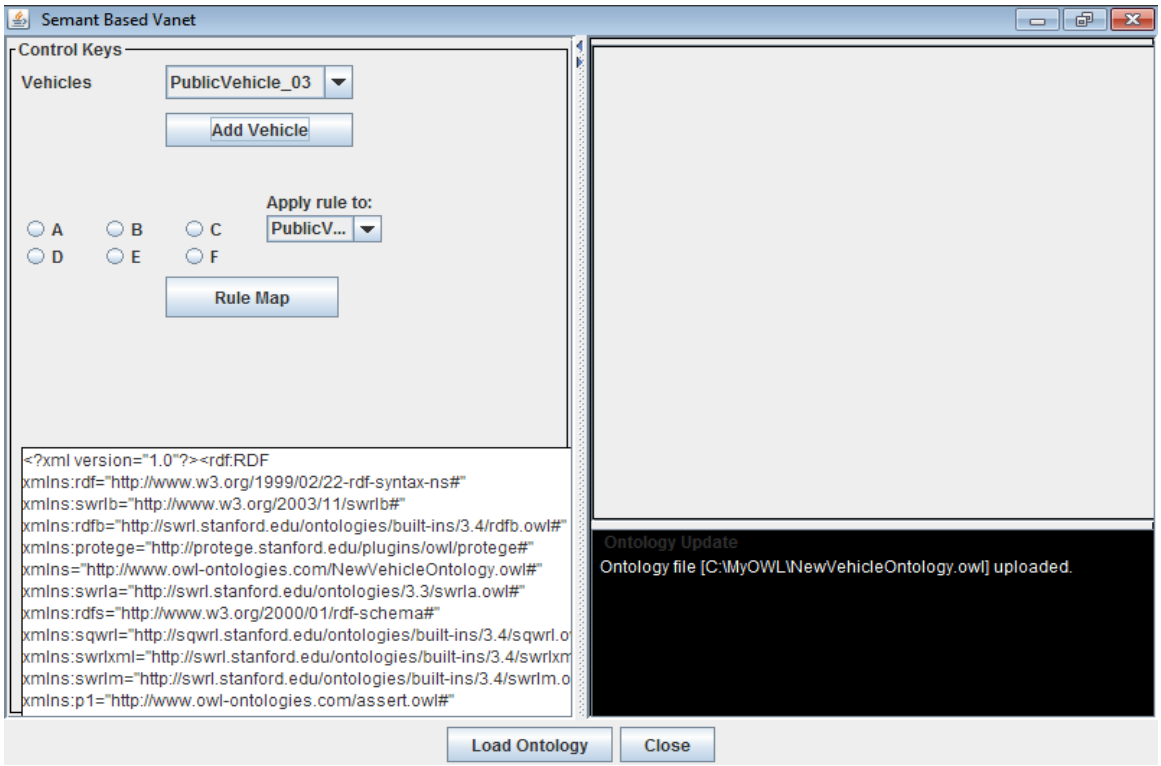


Figure 4.12 Outcome of Key Activation in the Two Fold Panels GUI showing Ontology Uploaded

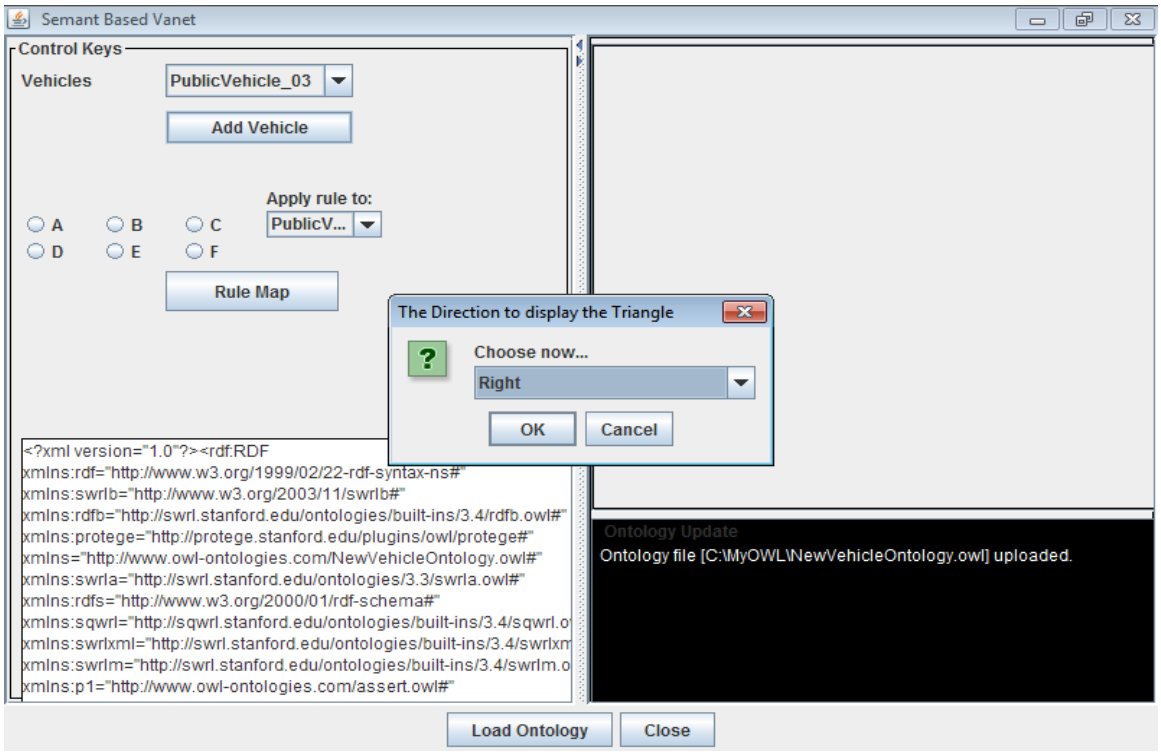


Figure 4.13 Outcome of Key Activation in the Two Fold Panels GUI showing Choice of Lane to Insert Vehicle

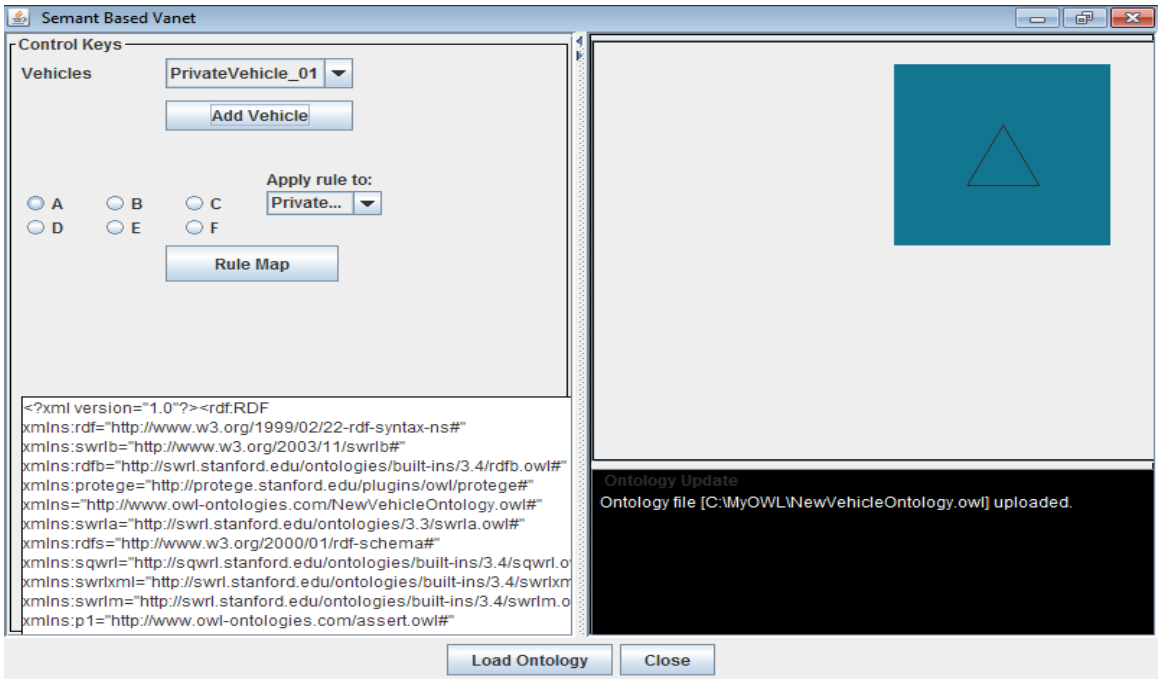


Figure 4.14 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted

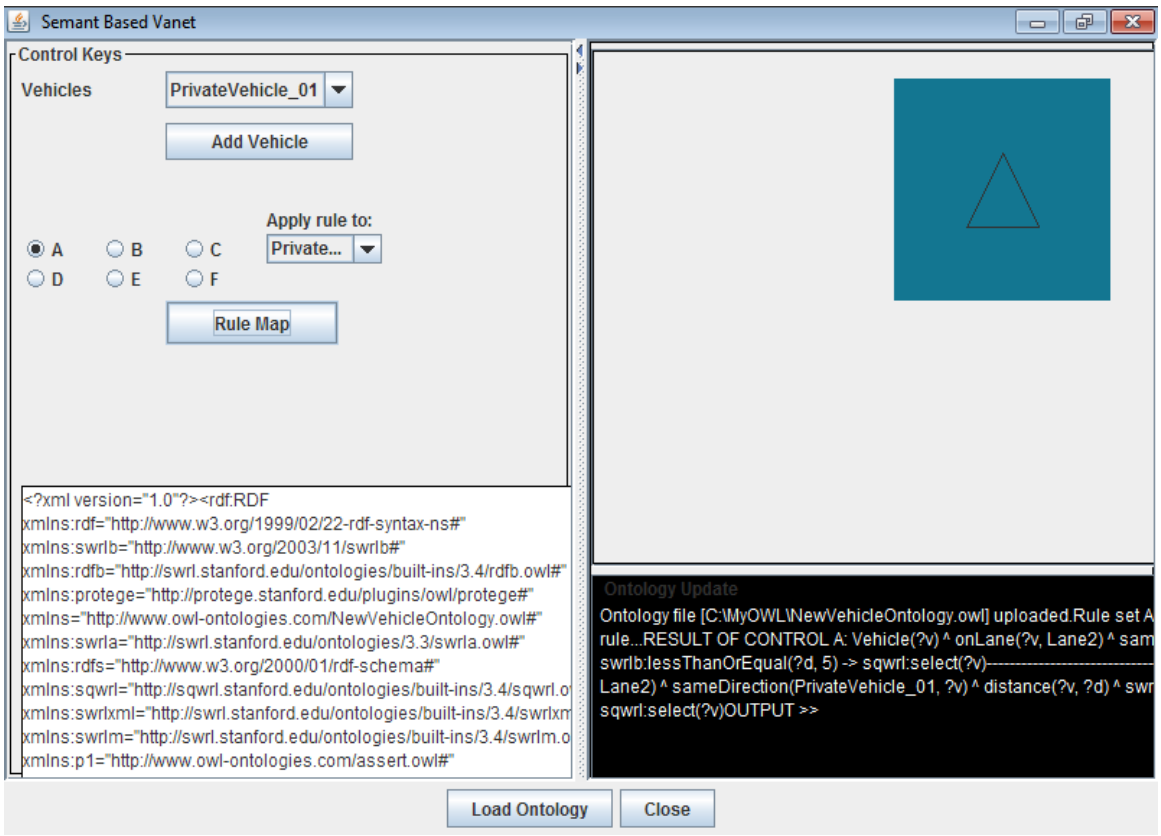


Figure 4.15 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (A) Applied

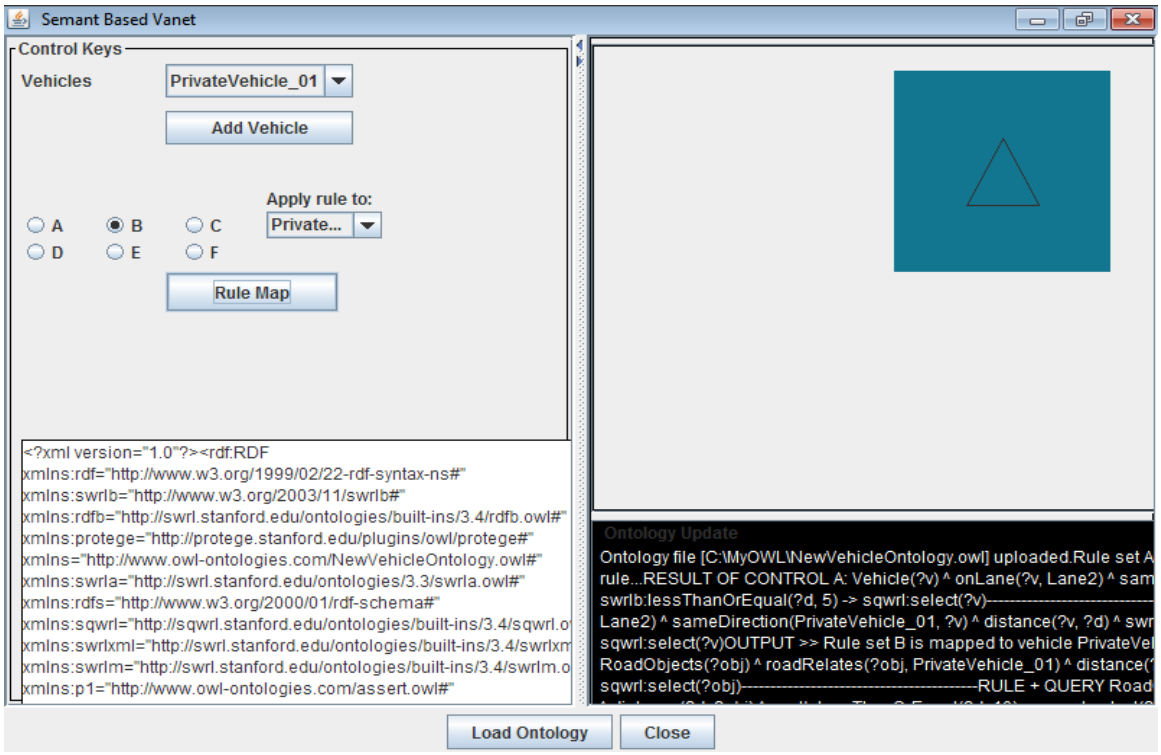


Figure 4.16 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (B) Applied

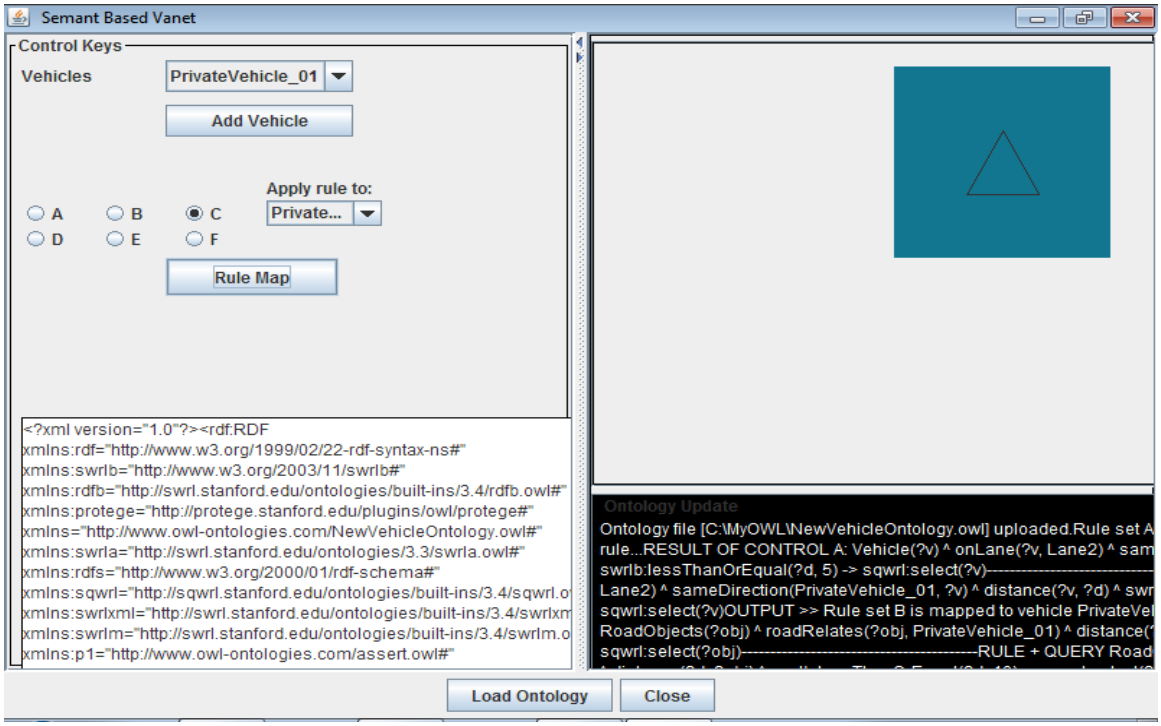


Figure 4.17 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (C) Applied

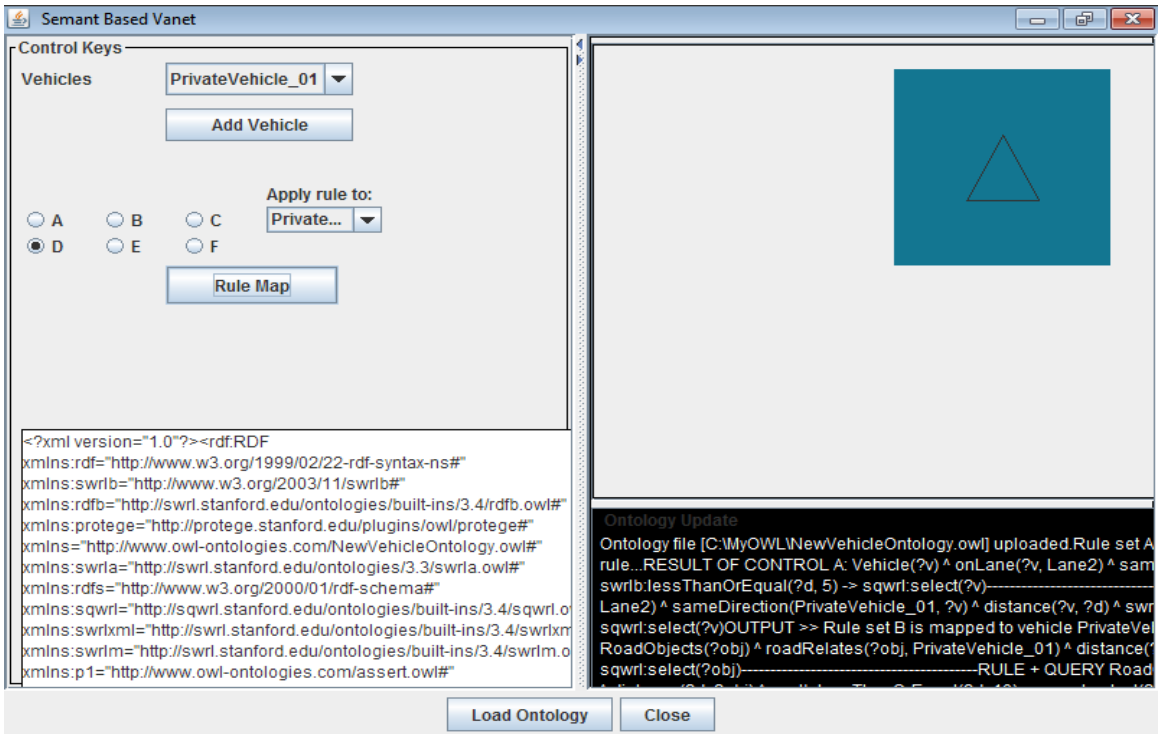


Figure 4.18 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (D) Applied

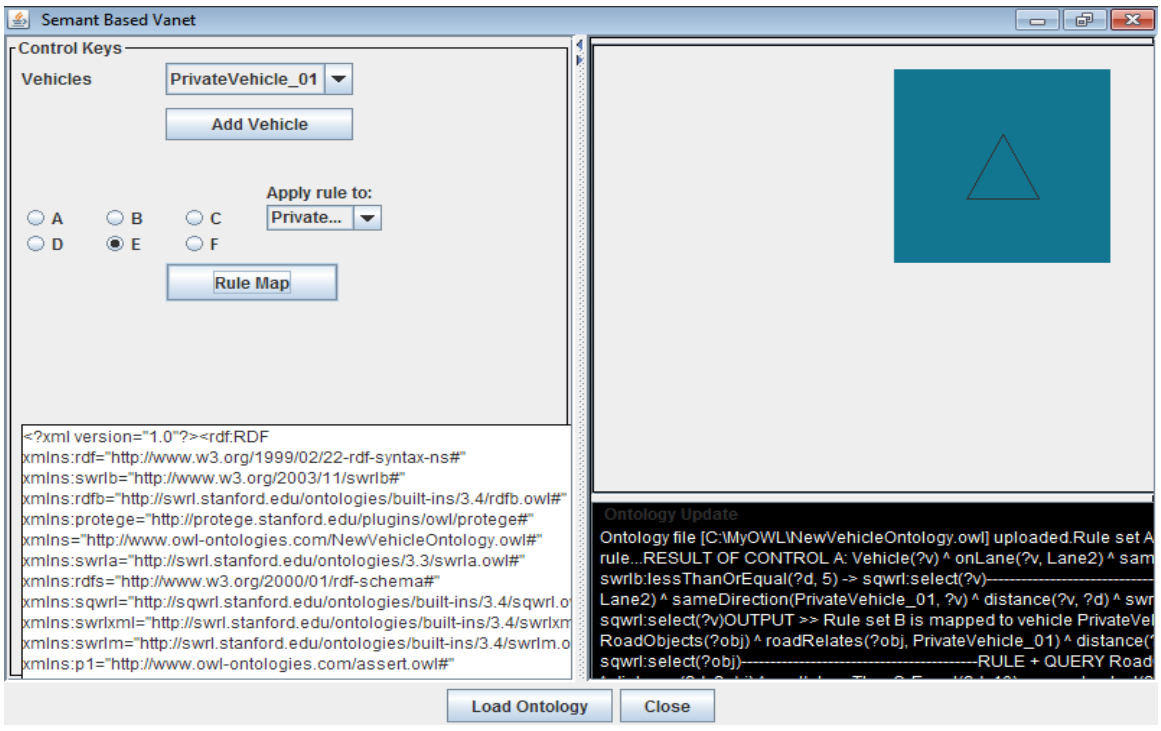


Figure 4.19 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (E) Applied

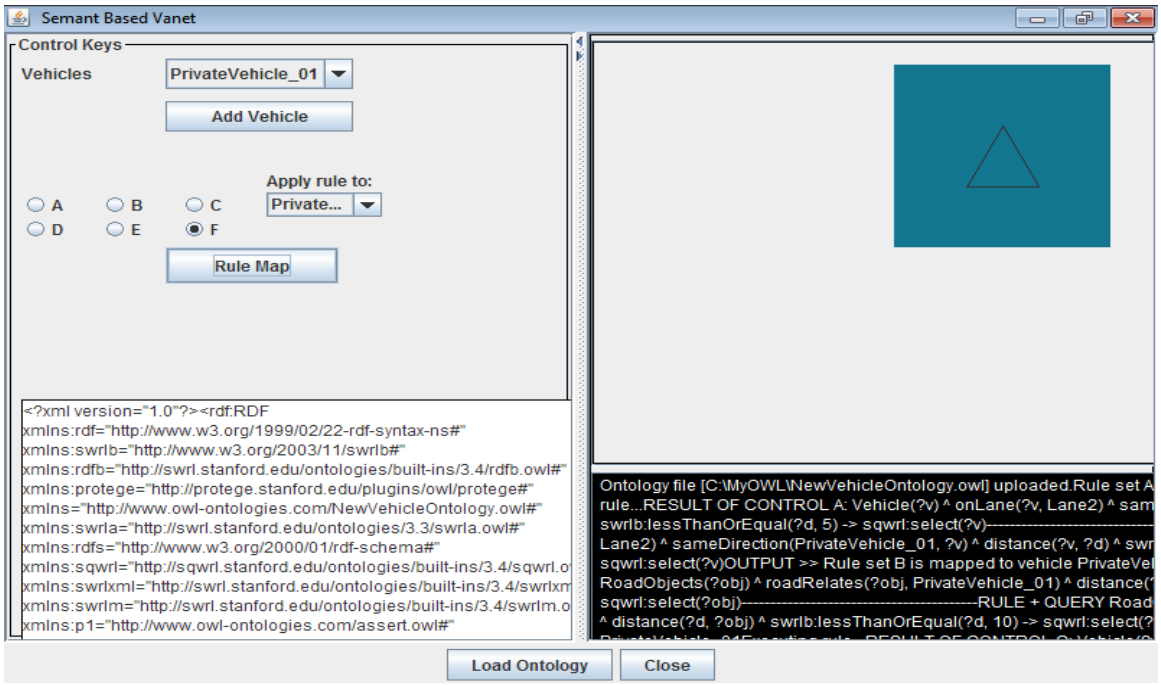


Figure 4.20 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted with Rule (F) Applied

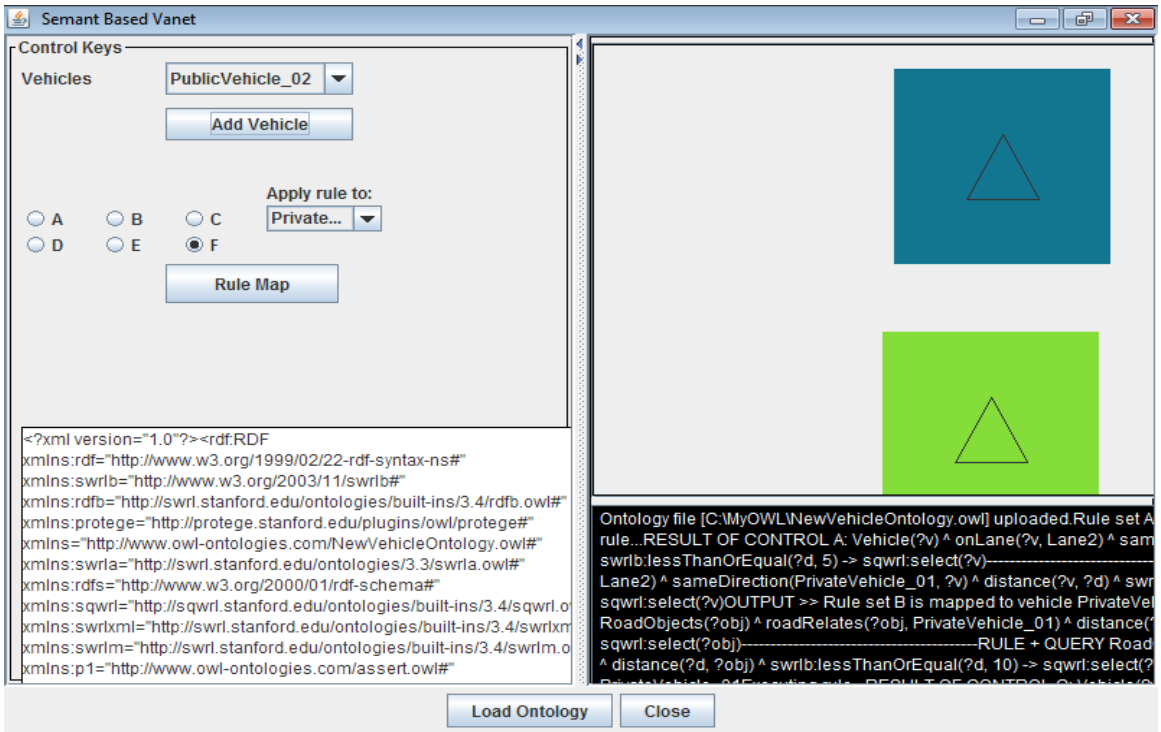


Figure 4.21 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted into Right Lane

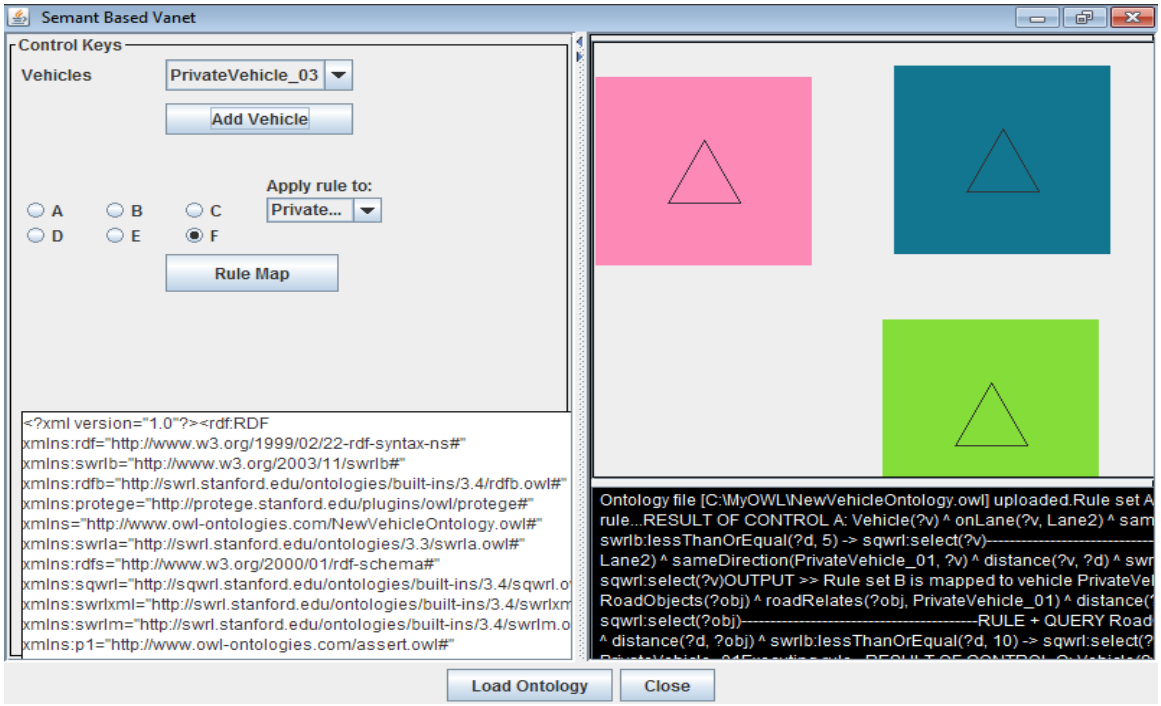


Figure 4.22 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted into Left and Right Lanes

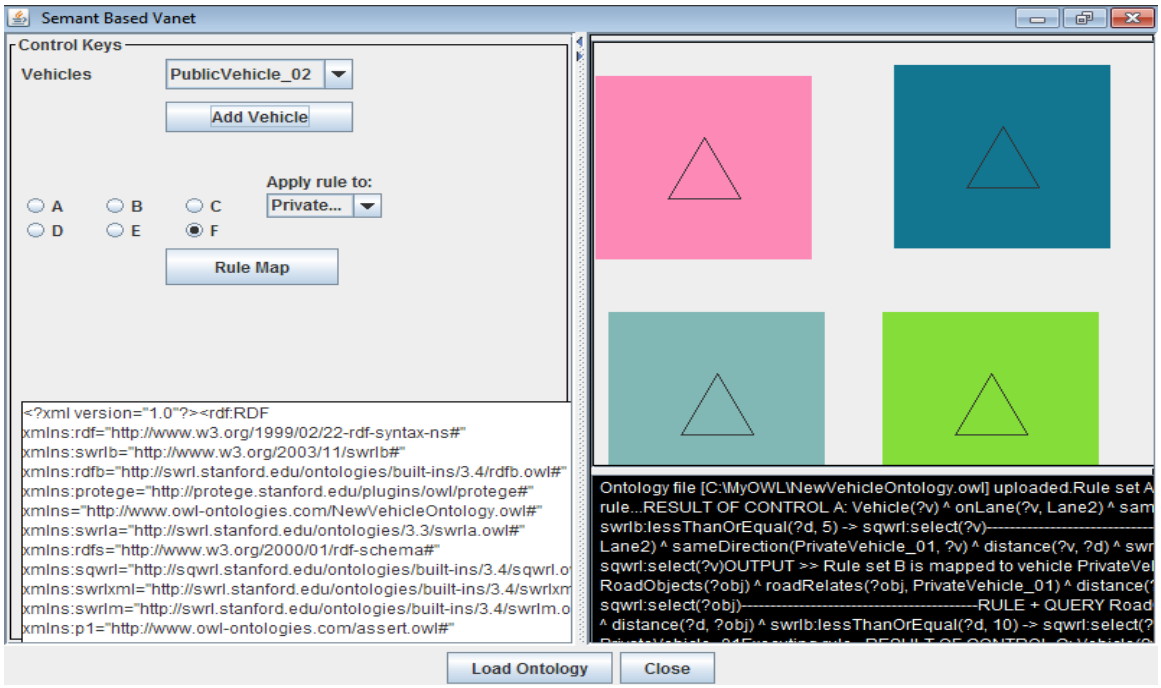


Figure 4.23 Outcome of Key Activation in the Two Fold Panels GUI showing Vehicle Inserted into Left and Right Lanes

4.6 Comparison with closest Work

This section compares the performance of the proposed system with the closest previous work from the literature review in terms of the capabilities of the system. The comparison is shown in Table 4.1.

Table 4.1: Comparison of Proposed System with some Previous Closest Work

Technology used		Morignot and Nashashibi, (2012)	Armand *et al.* *(*2014)	Proposed system
ADAS		Yes	Yes	Yes
ADAS&Human		No	No	Yes
Communication type design for the ontology	Leading vehicle	No	Yes	Yes
	SameDirection vehicle	No	Yes	Yes
	OppositeDirection vehicle	No	No	Yes
	RoadSideInfrastructure	No	Yes	Yes
	Pedestrian	Yes	Yes	Yes

Table 4.2: Comparison of Proposed System with some Previous Closest Work Modified

Technology used		Morignot and Nashashibi, (2012)	Armand *et al. (*2014)	Proposed system
ADAS		1	1	1
ADAS&Human		0	0	1
Communication type design for the	Leading vehicle	0	1	1
	SameDirection vehicle	0	1	1
	OppositeDirection vehicle	0	0	1
	RoadSideInfrastructure	0	1	1
	Pedestrian	1	1	1

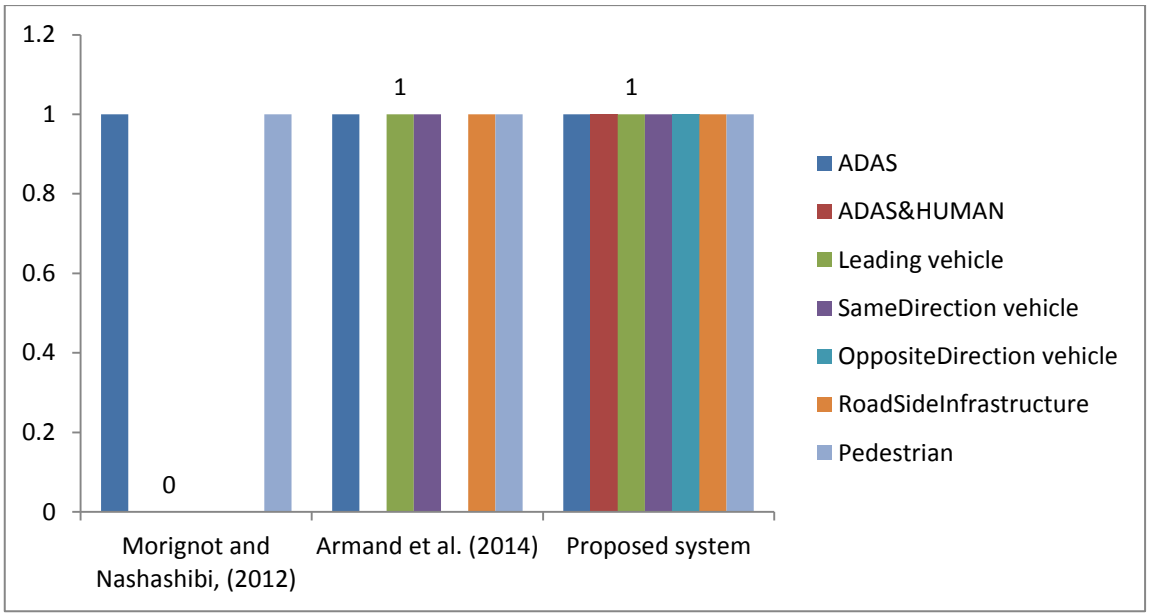


Figure 4.24 Comparison of the Proposed System with Some Previous Closest Work

Using the concept of electronics where 0 represent OFF and 1 represent ON, the Figure 4.24 shown the comparison of the proposed system with some closest previous work where YES is represented with 1 and NO represented with 0. The graph shows that the proposed system considered communication of ADAS and HUMAN and OppositeDirection which other works did not have, which make the proposed work to have more facilities compared to the previous works.

4.7 Discussion

Table 4.1 shows that the Advance Driving Assistance System (ADAS) can only communicate with their transportation agent.

Morignot and Nashashibi, and Armand *et al.*'s work, have ADAS facility, however in this research work ADAS is provided for vehicle usage in VANET system. Moreover, because Morignot and Nashashibi, and Armand *et al.*'s work are completely automated, there was no mentioning of the role of a human driven vehicle in their system meanwhile this work accommodated the role of a human.

Morignot and Nashashibi, works never consider the issues of leading vehicle – which is the communication between a vehicle and the vehicle ahead of it. Meanwhile this work and Armand *et al.* work added the concept of leading vehicle. In addition, Samedirection-ness movement of vehicles was omitted in Morignot and Nashashibi's work, while Armand *et al.* and this work considered Samedirection motion. Oppositedirection is another facility which was not provided in Morignot and Nashashibi, and Armand *et al.'s* work but was considered in this work. Oppositedirection enable vehicle to communicate with the vehicle coming from the opposite direction. RoadSideInfrastructure was not considered in Morignot and Nashashibi's work, while Armand *et al.* and this work considered the communication. Pedestrian facility was considered by Morignot and Nashashibi, and Armand *et al.* and was also included in this work. As explained in figure 4.24 the proposed system is more capable in terms of facilities it provided.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

This research work is able to realize the aim and objectives proposed. The aim is to improve performance in VANET system through the development and implementation of an enriched VANET ontology knowledge base while objectives are to develop an ontology-based VANET knowledge base. Rule sets (inference) meant to aid were provided to enhance reasoning capability and to implement a communication channel between VANET and its ontology knowledge based. The result has shown that when control key was activated for the ontology load view, output view for vehicle and ontology update view GUI, there was information sharing among participating vehicles.

5.2 Conclusion

In conclusion, this research has focused on improving the communication pattern among participating vehicles in a VANET system. This research achieved this by first, ontologically modeling the knowledge representation of the participating vehicles. Secondly, it designed and implemented a coordinated rule system for knowledge inference making. Lastly, it created a model of information exchange among participating vehicle.

The inter vehicle communication will improve the transportation system in Nigeria road situation with regard to congestion, accident and road safety. Vehicles can conveniently share information about their source, destination and location. This could be an added advantage in the case of car theft.

5.3 Recommendation for Future Work

In order to extend this work some other aspect which can be added to improve its capability are as follows:

a. The creation of more rules sets.
b. The addition of more vehicles to the Graphical User Interface to further enrich its functionalities.
c. Vehicle and the VANET system can be joined together to work with road side objects.
d. Rather than using a shape like triangle for modeling vehicle, a more realistic vehicle model can be used.

REFERENCES

Abburu, S. (2012): A Survey on Ontology Reasoners and Comparison: *International Journal of Computer Applications (0975 – 8887)* ***57*** *(17):* 33 – 39.

Armand, A; Filliat, D. and Ibañez-Guzman, J. (2014): Ontology-Based Context Awareness for Driving Assistance Systems. *Intelligent Vehicles Symposium Proceedings*, 2014 IEEE Dearborn, MI. 8-11 June 2014, 14452153: Pp227 – 233.

Arp´ırez, J. C; Corcho, O; Fern´andez-L´opez, M. and G´omez-P´erez, A. (2001): "WebODE: A Scalable Workbench for Ontological Engineering", K-CAP '01 *Proceedings of the 1st international conference on Knowledge capture* ACM New York, NY, USA ©2001 ISBN:1-58113-380-4 doi>10.1145/500737.500743 pp 6-13.

Bechhofer, S; Horrocks, I; Goble, C. and Stevens, C. (2001): "OILEd: A Reason-able Ontology Editor for The Semantic Web", KI2001, Joint German/Austrian conference on Artificial Intelligence, Springer-Verlag, Vienna. pp. **2174**: 396–408.

Bello, B. A. (2014): Extending an Ontology Alignment System with a Lexical Database. Unpublished M.Sc. thesis Department of Mathematics Ahmadu Bello University, Zaria pp 1 – 82.

Bermejo, A. J; Villadangos, J. and Astrain,J. J. (2013) "Ontology Based Road Traffic Management," Fortino *et al*,.(Eds): Intelligent distributed computing, VI, SCI 446, © Springer – Verlag Berlin Heidelberg 2013, ISSN: 1860-949X. Pp. 103–108.

Bernaras, A; Laresgoiti, I and Correra, J. (1996): *Building and Reusing Ontologies for Electrical Network Applications* ECAI96. 12th European conference on Artificial Intelligence. Ed. John Wiley & Sons, Ltd. Pp. 298-302.

Biswal, S. K. (2014): On Board Unit Based Authentication for V2V Communication in VANET. M.Tech. Thesis Department of Computer Science and Engineering National Institute of Technology Rourkela – 769008, India. Pp 1 – 46.

Boyce, S., and Pahl, C. (2007): Developing Domain Ontologies for Course Content. *Educational Technology & Society*, **10** (3), 275-288.

Brusa, G; Caliusco, M. L. and Chiotti O. (2006): Process for Building a Domain Ontology: an Experience in Developing a Government Budgetary Ontology Australian Computer Society, Inc. Paper appeared at the Australasian Ontology Workshop (AOW 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, M. A. Orgun and T. Meyer, Eds. **72**: 3 – 12.

Dean, G.S; Bechhofer, S; Harmelen, F. V; Hendler, J; Horrocks, I; McGuinness, D. L; Patel-Schneider, P. F. and Stein, L. A. (2004): *OWL Web Ontology Language Reference*, ed. W3C. Retrieved on 3th September 2014.

Dentler, K; Cornet, R; Teije, A. T. and Keizer, N de (2011): "Comparison of Reasoners for Large Ontologies in the OWL 2 EL Profile", *Semantic Web Journal*, pp.1-5.

Erritali, M; EL-Ouahidi, B; Hssina, B; Bouikhalene, B; and Merbouha, A. (2013): An Ontology-Based Intrusion Detection for Vehicular Ad Hoc Networks: *Journal of Theoretical and Applied Information Technology* 31st July 2013. **53** (3): 410 – 414.

Farquhar, A; Fikes, R. and Rice, J. (1997): "The Ontolingua Server: A Tool For Collaborative Ontology Construction", *International Journal of Human-Computer Studies* **46** (6): 707 - 727.

Fensel, D; Harmelen, F. V; Horrocks, I. McGuinness, D. L. and Patel-Schneider, P. F. (2001): *OIL: An ontology infrastructure for the semantic web.* IEEE Intelligent Systems. **16** (2) pp 38 – 45.

Gruber, T. R. (1993): *A translation Approach to portable ontology specifications. In Knowledge Acquisition*. **5**: 199-220.

H¨ulsen, M; Z¨ollner, J. M. and Weiss, C. (2011) "Traffic Intersection Situation Description Ontology for Advanced Driver Assistance," in Intelligent Vehicles Symposium, no. Iv, pp. 993–999.

Haarslev, V; Hidde, K; Moller, R. and Wessel, M. (2011): "The RacerPro Knowledge Representation and Reasoning Systems", *Journal Semantic Web IOS Press Amsterdam, The Netherlands, The Netherlands* **3** (3): 267-277.

Horrocks, I. (2002): *DAML+OIL: A Description Logic for the Semantic Web.* The IEEE Computer Society Technical Committee on Data Engineering. **25**(1): 4-9.

Horrocks, I; Patel-Schneider, P. F. and Harmelen, F. V. (2003): *From SHIQ and RDF to OWL:The Making of a Web Ontology Language. Journal of Web Semantics*. **1**(1):7-26.

Horrocks, I; Patel-Schneider, P. F; Boley, H; Tabet, S; Grosof, B and Dean, M. (2004): A Semantic Web Rule Language Combining OWL and RuleML Online. from SWRL, www.w3.org/Submission/SWRL/ SWRL: W3C Member Submission 21 May 2004. Access on 13th November, 2014.

http://swordfish.rdfweb.org/rdfquery/ online visited on (3th September, 2014): Reasoning Web: First International Summer School 2005, Msida, Malta. Edited by Eisinger, N. and Maluszynski, J.

http://www.mphasis.com: Introduction to Semantic Web", Retrieved Nov. 3, 2014 from http://www.mphasis.com/knowledge-center/white-papers-all.asp.

Hu, Y.-J; Yeh, C.-L. and Laun W. (2009): Challenes for Rule System on the Web. Rule Interchange and Applications: Proceedings International Symposium RuleML Las Vegas, Nevada, USA, November 5-7. Page 4 – 16.

Jain, V. and Singh, M. (2013): Ontology Based Information Retrieval in Semantic Web: A Survey. *I.J. Information Technology and Computer Science,* **10**: 62-69.

Kao, J. (2004): Wireless vehicular networks. Retrieved Nov. 3, 2014 from: http://www.cs.nthu.edu.tw/~jungchuk/research.html visited

Kazakov, Y; Krötzsch, M; and Simančík, F. (2012): "ELK Reasoner: Architecture and Evaluation", In Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012). *CEUR Workshop Proceedings*. Pp 106 – 117.

Kirthiga, N. and Karthik S. (2014): Vehicular Traffic Re-Routing for Avoid the Traffic Congestion in VANET. International Conference on Engineering Technology and Science-(ICETS'14) On 10th & 11th February Organized by Department of Civil, CSE, ECE, EEE, Mechnical Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India. www.ijirset.com pp 1530 – 1533.

Kiryakov, A. and Damova, M. (2011): "Reasoning in the Semantic Repositories", Handbook of Semantic Web Technologies, Springer. pp. 245-258.

Klyne, G. and Carroll, J. (2004): *Resource Description Framework (RDF): Concepts and Abstract Syntax.* W3C. Accessed Nov. 13, 2014 from *www.w3.org/TR/2004/REC-rdf-concepts-20040210/* W3C Recommendation

Manola F. and Miller, E. (2004): *RDF Primer*. W3C. Accessed Nov. 13, 2014 from *www.w3.org/TR/2004/REC-rdf-concepts-20040210/* W3C Recommendation pp 1 – 16.

McBride, B. (ed.) (2004): *RDF Test Cases.* W3C. Brickley, D. and Guha, R. V. (ed) (2014). Accessed Nov. 13, 2014 from http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

Mihoubi, H; Simonet, A. and Simonet, M. (2000): "An Ontology Driven Approach to Ontology Translation", In Proceedings of DEXA, pp.573-582.

Mishra, R. B. and Kumar, S. (2010): "Semantic Web Reasoners and Languages", Springer, DOI 10.1007/s10462-010-9197-3. Artificial Intelligence. **35** (4): 339-368.

Mohan, R. A. and Arumugam, G. (2011): Constructing Railway Ontology using Web Ontology Language and Semantic Web Rule Language. *International Journal of Computer Technology Application*, ISSN: 2229-6093 **2** (2): 314-321.

Morignot, P. and Nashashibi, F. (2012): An Ontology-Based Approach to relax Traffic Regulation for Autonomous Vehicle Assistance. INRIA Rocquencourt, Team IMARA, Domaine de Voluceau, B.P. 105, 78153 Le Chesnay, France. Pp 1 – 8.

Nair, V. and Dutta, A. (2010):"Ontology based Session Management Protocol for Teleteaching Domain", Proceedings of the 2nd International Conference on Computer and Automation Engineering", **5**: 866-870.

Neches, R; Fikes, R; Finin, T; Gruber, T; Patil, R; Senator, T. and Swartout, W.R. (1991): *Enabling Technology for Knowledge Sharing*. AI Magazine. Pp. 36-56.

Noy, N. F and McGuinness, D. L, (2001): "Ontology Development 101: A Guide To Creating Your First Ontology", Stanford Medical Informatics Technical Report SMI-2001-0880, pp 1 – 25.

Obiniyi, A. A.; Oyelade, O. N. and Auta G. I. (2014): Ontology Languages, Development Tools and Repositories: Towards a Unifying Platform. *International Journal of Computer Science and Artificial Intelligence* **4** (3): 68 – 74.

Pandey, G. (2012): The Semantic Web: An Introduction and Issues: International *Journal of Engineering Research and Applications (IJERA)* ISSN: 2248-9622 www.ijera.com **2** (1): 780-786.

Pollard, E; Morignot, P. and Nashashibi, F. (2013): An Ontology-based Model to Determine the Automation Level of an Automated Vehicle for Co-Driving. Published in16th International Conference on Information Fusion. hal-00838680, version 1 – 27.

Ratnakar, V. and Gil, Y. (2002): *A Comparison of Markup Languages.* Proceedings of the *15th International* Florida Artificial Intelligence Research Society *Conference*, FLAIRS Florida Artificial Intelligence Research Symposium *FLAIRS Conference* Menlo Park, California, USA, AAAI Press, (*2002*). *page* 413-418.

Saravanan, K; Arunkumar, T. and RameshBuba, K. (2010): An Intelligent Driver Assistance System (I-DAS) for Vehicle Safety Modelling using Ontology Approach: International Journal Of UbiComp (IJU), **1** (3): 15 – 29.

Sattler U, (2007): "Description Logic Reasoners", http://www.cs.man.ac.uk/~sattler/ reasoners.html, website accessed on Feb 19, 2007.

Seaborne, A. and Prud'hommeaux, E. (2006): *SPARQL Query Language for RDF. W3C Candidate Recommendation 6 April 2006 Accessed Nov. 13, 2014.http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/.*

Seddiqui, M. H. and Aono, M. (2010): Metric of Intrinsic Information Content for Measuring Semantic Similarity in an Ontology. Proc. 7th Asia-Pacific Conference on Conceptual Modelling (APCCM 2010), Brisbane, Australia. Pp. 89 – 96.

Shastri, A; Dadhich, R. and Poonia R. C. (2011): Performance Analysis of On-Demand Routing Protocols for Vehicular ad-hoc Networks. International Journal of Wireless & Mobile Networks (IJWMN) **3** (4): 103 – 111.

Shearer, R; Motik, B. and Horrocks, I. (2008): HermiT: A Highly-Efficient OWL Reasoner. Oxford University Computing Laboratory Oxford, OX1 3QD, UK. Pp 1 – 10.

Sirin, E; Parsia, B; Grau, B. C; Kalyanpur a, A; and Katz, Y. (2007): Pellet: A Practical OWL-DL Reasoner. University of Maryland, MIND Lab, 8400 Baltimore Ave, College Park MD 20742, USA. Web Semantics: Science, Services and Agents on the World Wide Web, **5** (2): 51–53.

Smith, M. K; Welty, C. and McGuinness, D. L. (2004): *OWL Web Ontology Language Guide.* W3C, Recommendation. Accessed Nov. 13 2013. http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

Standard, and American Society for Testing and Materials (2003) "E2213-03, "Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems" 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications". Pp 104 – 110.

Studer, R; Benjamins, V. R. and Fensel, D. (1998): Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*. **25**: 161-197.

Sure, Y. (2002): "On-To-Knowledge: Ontology Based Knowledge Management Tools and Their Application", *German Journal Kuenstliche Intelligenz*, Special Issue on Knowledge Management. Pp 35 – 37.

Sure, Y; Erdmann, M; Angele, J; Staab, S; Studer, R. and Wenke, D. (2002): "OntoEdit: Collaborative Ontology Engineering For The Semantic Web". Proceedings of the First International Semantic Web Conference, Sardinia, Italia. Pp 221 – 235.

Swartout, B; Patil, R; Knight, K. and Russ, T. (1997): *Toward Distributed Use of Large-Scale Ontologies* Ontological Engineering. AAAI-97 Spring Symposium Series. Pp. 138-148.

SWRL-Submission, http://www.w3.org/submission/SWRL. Accessed Nov. 13 2014.

Tarik, T; Ehssan, S. and Abbas. J. (2007): "A Stable Routing Protocol to Support ITS Services in VANET Networks", *IEEE Transactions on Vehicular Technology,* **56**(6): 3337- 3347.

The Prot´eg´e project, http://protege.stanford.edu. Accessed Nov. 13 2014.

Uschold, M. and Gruninger, M. (1996): Ontologies: Principles, Methods and Applications. © University of Edinburgh 1996. **11** (2): 1 – 69.

Verma, A. and Gujral, M. S. (2012): An Ontological View of Trusted OLSR Protocol of ad hoc Network. *Special Issue of International Journal of Computer Applications on Issues and Challenges in Networking, Intelligence and Computing Technologies – ICNICT 2012. (0975 – 8887) Pp 18 – 21.*

Williams H. (2013): 5,000 WORD ESSAY – 2001: A Lagos odyssey past and future: http://www.mob76outlook.com/5000-word-essay-2001-a-lagos-odyssey/ accessed on 4th Nov. 2014.

Wooldridge, M. (2009): An Introduction to Multi Agent Systems-Second Edition. *Department of Computer Science, University of Liverpool, UK* John Wiley & Sons T&T Productions Ltd, London. Printed and bound in Great Britain by Biddies Ltd, Guildford and Kings Lynn. Pp 1 – 365.

Xiaodong, L; Sun, X; Ho, P. H. and Shen, X. (2007): "GSIS: a secure and privacy-preserving protocol for vehicular communications." Vehicular Technology, IEEE Transactions **56** (6): 3442-3456.

Zhang, Z. (2005): Ontology Query Languages for the Semantic Web: A Performance Evaluation. M.Sc. thesis of The University of Georgia, Athens, Georgia. Pp 1 – 57.

Zhu, J. and Roy, S. (2003): MAC for dedicated short range communications in intelligent transport system. Communications Magazine, IEEE 7950580 IEEE Communications Society Published: IEEE. **41** (12): 60 – 67.

APPENDIX

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package semanticapp;

/**


 *
 * @author Sikirat
 */
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Component;
import java.awt.GridLayout;
import java.awt.event.*;
import java.awt.event.*;
import java.util.ArrayList;
import javax.swing.BorderFactory;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JSplitPane;
import javax.swing.JFileChooser;
import javax.swing.border.TitledBorder;
import java.io.*;
import java.util.Random;

/**
 *
 * @author Sikirat
 */
public class Main22 extends JFrame implements ActionListener, ItemListener {

    /**
     * @param args the command line arguments
     */
int sn = 0;
int sn1 = 0;
```

```
int sn2 = 0;
int ONT_LOAD=0;
JavaTriangle triangle = null;
String input = "";

ArrayList ruleMapV1=new ArrayList();
ArrayList ruleMapV2=new ArrayList();
ArrayList ruleMapV3=new ArrayList();
ArrayList ruleMapV4=new ArrayList();

JPanel panelwest = new JPanel();
JPanel paneleast = new JPanel();
JPanel panelsouth = new JPanel();
JPanel side1 = new JPanel();
JPanel side2 = new JPanel();
JTextArea notepad=new JTextArea();
JTextArea output_cmd=new JTextArea();
JButton loadont = new JButton("Load Ontology");
JButton close = new JButton("Close");
JButton simulate = new JButton("Simulate");
JButton keymap = new JButton("Rule Map");
JButton addvehicle = new JButton("Add Vehicle");
JButton keymapb = new JButton("Rule Map");
JLabel vehiclelabel = new JLabel("Vehicles");
JLabel canvaslabel = new JLabel("Canvas");
JLabel vehiclecombo2lbl = new JLabel("Apply rule to:");
JLabel blocklabel = new JLabel("CMD or Black Background \t \t");
String veh[];//={"Choose One","VO 1","VO 2","VO 3","VO 4"};
JComboBox vehiclecombo;// = new JComboBox(veh);
static String currentVehicle;
JComboBox vehiclecombo2 = new JComboBox();
ButtonGroup bg = new ButtonGroup();
JRadioButton ab = new JRadioButton("A");
JRadioButton bb = new JRadioButton("B");
JRadioButton cb = new JRadioButton("C");
JRadioButton db = new JRadioButton("D");
JRadioButton eb = new JRadioButton("E");
JRadioButton fb = new JRadioButton("F");
TitledBorder title1 = null;
TitledBorder title2 = null;
TitledBorder title3 = null;

RuleImplementer rm=new RuleImplementer();
JPanel pan = new JPanel();
   public Main22(){
      super("Semant Based Vanet");
      setLayout(new BorderLayout());
      panelwest.setLayout(null);
     // panelwest.setSize(250,300);
```

```
output_cmd.setBorder(BorderFactory.createTitledBorder(BorderFactory.createLineBor
der(Color.black),""));
        output_cmd.setWrapStyleWord(true);
        output_cmd.setLineWrap(true);

        title1 =
BorderFactory.createTitledBorder(BorderFactory.createLineBorder(Color.black),"Contr
ol Keys");
        title2 =
BorderFactory.createTitledBorder(BorderFactory.createLineBorder(Color.black),"");
        title3 =
BorderFactory.createTitledBorder(BorderFactory.createLineBorder(Color.black),"Ontol
ogy Update");
        title1.setTitleJustification(TitledBorder.LEFT);
        panelwest.setBorder(title1);
        paneleast.setBorder(title2);
        vehiclelabel.setBounds(10, 20, 100, 25);


        veh=rm.getOWLInstances();
        vehiclecombo = new JComboBox(veh);
        vehiclecombo.addItemListener(this);
        vehiclecombo.setBounds(110, 20, 130, 25);
        addvehicle.setBounds(110, 55, 130, 25);
       // panelwest.setSize(200, 200);
         bg.add(ab);
         bg.add(bb);
         bg.add(cb);
         bg.add(db);
         bg.add(eb);
         bg.add(fb);
         ab.setBounds(10,130,50,20);
         bb.setBounds(65,130,50,20);
         cb.setBounds(120,130,50,20);
         db.setBounds(10,150,50,20);
         eb.setBounds(65,150,50,20);
         fb.setBounds(120,150,50,20);
         vehiclecombo2lbl.setBounds(180,110,120,20);
         vehiclecombo2.setBounds(180,130,80,20);
         keymap.setBounds(110,175,120,30);
         simulate.setBounds(90,250,140,30);
         //panelwest.add(simulate);

         notepad.setBounds(10,300,470,300);

notepad.setBorder(BorderFactory.createTitledBorder(BorderFactory.createLineBorder(
Color.black),""));
         notepad.setWrapStyleWord(true);
         notepad.setLineWrap(true);
```

```
        panelwest.add(notepad);

        panelwest.add(vehiclecombo2lbl);
        panelwest.add(vehiclecombo2);
        panelwest.add(ab);
        panelwest.add(bb);
        panelwest.add(cb);
        panelwest.add(db);
        panelwest.add(eb);
        panelwest.add(fb);
       panelwest.add(keymap);
       panelwest.add(vehiclelabel);
       panelwest.add(vehiclecombo);
       panelwest.add(addvehicle);
       side1.setBorder(title2);
       //side2.setBorder(title3);
       output_cmd.setBorder(title3);
       panelsouth.add(loadont);
       panelsouth.add(close);
       canvaslabel.setBackground(Color.BLUE);
       paneleast.setLayout(null); //new BorderLayout());
       pan.setLayout(null);
       side1.setLayout(new BorderLayout());

       side2.setLayout(new BorderLayout());
       output_cmd.add(blocklabel,BorderLayout.CENTER);
       output_cmd.setBackground(Color.black);
       output_cmd.setForeground(Color.white);

       JScrollPane scrollPane1 = new JScrollPane(side1);

scrollPane1.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALW
AYS);
       scrollPane1.setBounds(1, 5, 700, 350);
       paneleast.add(scrollPane1); //,BorderLayout.CENTER);
       JScrollPane scrollPane2 = new JScrollPane(output_cmd);
       //scrollPane2.setMaximumSize(new Dimension(300, 300));

scrollPane2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALW
AYS);
       scrollPane2.setBounds(1, 360, 700, 340);
       paneleast.add(scrollPane2);//,BorderLayout.SOUTH);

       JScrollPane west = new JScrollPane(panelwest);
       JScrollPane east = new JScrollPane(paneleast);
       JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, true,
west, east);
      //System.out.println("The size of Split "+splitPane.getHeight());
       splitPane.setResizeWeight(0.5f);
       splitPane.setDividerLocation(.5f);
```

```
   loadont.addActionListener(this);
   close.addActionListener(this);
   simulate.addActionListener(this);
   keymap.addActionListener(this);
   addvehicle.addActionListener(this);
   keymapb.addActionListener(this);
   splitPane.setOneTouchExpandable(true);
   getContentPane().add(splitPane,BorderLayout.CENTER);
   getContentPane().add(panelsouth,BorderLayout.SOUTH);
   setSize(1200,700);
   setVisible(true);
   setLocationRelativeTo(null);
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

}
public static void main(String[] args) {
   // TODO code application logic here
   new Main22();
}

public void itemStateChanged(ItemEvent e)
{
 if(e.getStateChange() == ItemEvent.SELECTED)
   {
     currentVehicle=(String)vehiclecombo.getSelectedItem();
   }
}
public void actionPerformed(ActionEvent e)
{

   if(e.getSource() == loadont)
   {
     final JFileChooser fc = new JFileChooser();

     int returnVal = fc.showOpenDialog(this);

      if (returnVal == JFileChooser.APPROVE_OPTION)
      {
         File file = fc.getSelectedFile();
         String fileName=file.getAbsolutePath(); //+file.getName();
         Constants22.OWL_FILE+=fileName;
         //displayError(Constants.OWL_FILE);
         try
         {
           FileReader inputFile = new FileReader(fileName);
           BufferedReader bufferReader = new BufferedReader(inputFile);
           String line;
           while ((line = bufferReader.readLine()) != null)
           {
            notepad.append(line);
```

```
                }
                showCMD("Ontology file ["+fileName+"] uploaded.");
                ONT_LOAD=1;
                bufferReader.close();
            }
            catch(Exception ex)
            {
                System.out.println("Error while reading file line by line:" +
ex.getMessage());
            }
        }
        else
        {

    if(e.getSource() == keymap)
        {
            String[] setArules={"Vehicle(?v) ^ onLane(?v, Lane2) ^
sameDirection("+currentVehicle+", ?v) ^ distance(?v, ?d) ^ swrlb:lessThanOrEqual(?d,
5) -> sqwrl:select(?v)"};
                    //,"Vehicle(?v) ^ onLane(?v, Lane1) ^ sameDirection(currentVehicle,
?v) ^ distance(?v, ?d) ^ swrlb:lessThanOrEqual(?d, 3) -> sqwrl:select(?v)"};
            String[] setAvars={"?v"};

            String[] setBrules={"RoadObjects(?obj) ^ roadRelates(?obj,
"+currentVehicle+") ^ distance(?d, ?obj) ^ swrlb:lessThanOrEqual(?d, 10) ->
sqwrl:select(?obj)"};
            String[] setBvars={"?obj"};

            String[] setCrules={"Vehicle(?v) ^ isOnMotion(?v, Motion) ^ plies(?v, ?lane)
? sqwrl:select(?lane)"};
            String[] setCvars={"?lane"};

            String[] setDrules={"Vehicle(?v) ^ oppositeDirection("+currentVehicle+",
?v)  ? sqwrl:select(?v)"};
            String[] setDvars={"?v"};

            String[] setErules={"RoadObjects(?obj)  ^
abox:hasClass(TrafficControlSystem, ?obj) ^ currentSignal(?obj, ?signal) ^
averageSpeed(?signal, ?r) ? sqwrl:select (?r, ?signal)"};
            String[] setEvars={"?r, ?signal"};

            String[] setFrules={"Vehicle(?v) ^  pathRelates(?v, "+currentVehicle+") ^
distance(?v, ?d) ^ swrlb:lessThanOrEqual(?d, 5) ? sqwrl:select (?v)"};
             String[] setFvars={"?v"};
            if(vehiclecombo2.getItemCount() > 0)
            {
               String rules[]={""};
               String all_vars[]={""};

               try{
```

```
String choice="";
if(ab.isSelected()){
   choice = ab.getText();
   rules=setArules;
   all_vars=setAvars;
}else if(bb.isSelected()){
   choice = bb.getText();
   rules=setBrules;
   all_vars=setBvars;
}else if(cb.isSelected()){
   choice = cb.getText();
   rules=setCrules;
   all_vars=setCvars;
}else if(db.isSelected()){
   choice = db.getText();
   rules=setDrules;
   all_vars=setDvars;
}if(eb.isSelected()){
   choice = eb.getText();
   rules=setErules;
   all_vars=setEvars;
}if(fb.isSelected()){
   choice = fb.getText();
   rules=setFrules;
   all_vars=setFvars;
}

int v=vehiclecombo.getSelectedIndex();
output_cmd.setForeground(Color.red);
switch(v)
{
   case 0:
      if(ruleMapV1.contains(choice))
        {
         displayError("Rule " + choice + " is already applied to vehicle
V1");
         showCMD("Rule "+choice+" is already applied to vehicle V1");
        }
      else
        {
         ruleMapV1.add(choice);
        }
      break;

   case 1:
      if(ruleMapV2.contains(choice))
      {
         displayError("Rule " + choice + " is already applied to vehicle
V2");
         showCMD("Rule "+choice+" is already applied to vehicle V2");
```

```
                    }
                    else
                       ruleMapV2.add(choice);
                    break;

                 case 2:
                    if(ruleMapV3.contains(choice))
                    {
                       displayError("Rule " + choice + " is already applied to vehicle
V3");
                       showCMD("Rule "+choice+" is already applied to vehicle V3");
                    }
                    else
                       ruleMapV3.add(choice);
                    break;

                 case 3:
                    if(ruleMapV4.contains(choice))
                    {
                       displayError("Rule " + choice + " is already applied to vehicle
V4");
                       showCMD("Rule "+choice+" is already applied to vehicle V4");
                    }
                    else
                       ruleMapV4.add(choice);
                    break;
              }
              output_cmd.setForeground(Color.white);
              //System.out.println("Choice "+choice);
              showCMD("Rule set "+choice+" is mapped to vehicle
"+vehiclecombo.getSelectedItem());

              ArrayList<String> outputAresults=new ArrayList<String>();
              System.out.println("No 2");
              showCMD("Executing rule...");
              for(int n=0; n < rules.length; ++n)
              {
                 System.out.println("No 3");
                 String result=rm.queryWitSQWRL(rules[n], all_vars,
choice+"_Query_"+n);
                 String final_output="RESULT OF CONTROL "+choice+":
"+rules[n];
                 final_output+="----------------------------------------";
                 final_output+="RULE + QUERY "+rules[n];
                 final_output+="OUTPUT >> "+result;
                 System.out.println(final_output);
                 outputAresults.add(final_output);
                 showCMD(final_output);
                 new ThreadHandler(choice, n).start();
              }
```

```
                }
                catch(Exception ee)
                {
                   System.out.println("Error Ocured Due to "+ee.getMessage());
                }
             }
             else
             {
              displayError("No vehicle selected");  output_cmd.setForeground(Color.red);
              showCMD("No vehicle selected");
output_cmd.setForeground(Color.white);
             }

          }else if(e.getSource() == close){
             System.exit(0);
          }
          else {


         currentVehicle=(String)vehiclecombo.getSelectedItem();
           System.out.println("@@@@@@@@@@@ "+currentVehicle);
             if(ONT_LOAD > 0 ) {
                int x=0;
                int y=0;
                if(vehiclecombo.getSelectedIndex()==0){
                   JOptionPane.showMessageDialog(null, "Please Choose a Vehicle",
"Error", JOptionPane.ERROR_MESSAGE);
                }else{
                   String[] choices = { "Left", "Right"};
                   String input = (String) JOptionPane.showInputDialog(null, "Choose
now...",
                         "The Direction to display the Triangle",
JOptionPane.QUESTION_MESSAGE, null, // Use
                         // default
                         // icon

                         choices, // Array of choices
                         choices[1]); // Initial choice
                   vehiclecombo2.addItem(veh[vehiclecombo.getSelectedIndex()]);
                   JavaTriangle triangle = new JavaTriangle();
                   side1.removeAll();
                   // side1.setLayout(new BorderLayout());
                   pan.add(canvaslabel);
                   System.out.println("The choice Made is "+input);
                   Random rn = new Random();
                   if(input.equals("Left")){
                      sn1++;
                      if(sn1>2){
                         JOptionPane.showMessageDialog(null, "Maximum of Two
Vehicles are Allowed per Lane", "Error", JOptionPane.ERROR_MESSAGE);
```

```
    }else{
        Component[] components = pan.getComponents();
        Component component = null;
        for (Component component1 : components) {
            component = component1;
            if(component instanceof JavaTriangle){
                //your code here
                //sn1++;
            } }
        System.out.println("SNO1 "+sn1);
        if(sn1 >1){
            x = 10;
            y = 220;
        }else{
            x = rn.nextInt(15 - 1 + 1) + 1;
            y = rn.nextInt(50 - 20 + 1) + 20;
        }

        triangle.setBounds(x,y,150,150);
        triangle.setBackground(Color.blue);
        pan.add(triangle);
        pan.revalidate();
    }
}else if(input.equals("Right")){
    sn2++;
    Component[] components = pan.getComponents();
    Component component = null;
    for (Component component1 : components) {
        component = component1;
        if(component instanceof JavaTriangle){
            //your code here
            // sn2++;
        }  }
    System.out.println("SNO2 "+sn2);
    if(sn2 > 1){
        x = 200;
        y = 220;
    }else{
        x = rn.nextInt(210 - 190 + 1) + 190;
        y = rn.nextInt(20 - 10 + 1) + 10;
    }

    triangle.setBounds(x,y,150,150);
    triangle.setBackground(Color.blue);
    pan.add(triangle);
    pan.revalidate();
}
/**
 * JPanel panel = new JPanel
 * Component[] cpmponents = panel.getComponents();
```

```
                    * Component component = null;
                    * for(int i=0;i< components.length;i++){
                    * component = components[i];
                    * if(component instance of JTriangle){
                    * //your code here
                    * }
                    * }
                    *
                    */
                   //side1.setSize(100, 200);
                   Random rand = new Random();
int r = rand.nextInt(255);
int g = rand.nextInt(255);
int b = rand.nextInt(255);
Color randomColor = new Color(r, g, b);
           //side1.setSize(100, 200);
 side1.add(pan,BorderLayout.CENTER);
            triangle.setBackground(Color.GRAY);
            triangle.setBackground(randomColor);
           side1.revalidate();
         }
        }
        else
         {
           JOptionPane.showMessageDialog(null, "Please load ontology first", "Error",
JOptionPane.ERROR_MESSAGE);
         }
    }
      // throw new UnsupportedOperationException("Not supported yet.");
    }
    public void displayError(String choice)
    {
     JOptionPane.showMessageDialog(null, choice, "Error",
JOptionPane.ERROR_MESSAGE);
    }
    //Writing method to show Display here
    public void showCMD(String choice)
    {
       output_cmd.append(choice);
       /*String txt = "Action Key "+choice+" selected by user Vehicle
"+vehiclecombo.getSelectedItem().toString()+"has has changed lane";
       side2.removeAll();
       side2.add(new JLabel(txt),BorderLayout.CENTER);
       side2.revalidate();*/
    }
}
```