Towards Intelligent Transport Systems:

Geospatial Ontological Framework and Agent Simulation

Seong Kyu Choi

Thesis submitted for the Degree of Doctor of Philosophy (PhD)

Department of Civil, Environmental and Geomatic Engineering
UCL

2017

I, Seong Kyu Choi, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

In an Intelligent Transport System (ITS) environment, the communication component is of high significance as it supports interactions between vehicles and the roadside infrastructure. Existing studies focus on the physical capability and capacity of the communication technologies, but the equally important development of suitable and efficient semantic content for transmission has received notably less attention. Using an ontology is one promising approach for context modelling in ubiquitous computing environments. In the transport domain, an ontology can be used both for context modelling and semantic contents for vehicular communications. This research explores the development of an ontological framework implementing a geosemantic messaging model to support vehicle-to-vehicle communications.

To develop an ontology model, two scenarios (an ambulance situation and a breakdown on the motorway) are constructed to describe specific situations using short-range communication in an ITS environment. In the scenarios, spatiotemporal relations and semantic relations among vehicles and road facilities are extracted and defined as classes, objects, and properties/relations in the ontology model. For the ontology model, some functions and query templates are also developed to update vehicles' movements and to provide some logical procedures that vehicles need to follow in emergency situations. To measure the effects of the vehicular communication based on the ontology model, an agent-based approach is adopted to dynamically simulate the moving vehicles and their communications following the scenarios.

The simulation results demonstrate that the ontology model can support vehicular communications to update each vehicle's context model and assist its decision-making process to resolve the emergency situations. The results also show the effect of vehicular communications on the efficiency trends of traffic in emergency situations, where some vehicles have a communication device, and others do not. The efficiency trends, based on the percentage of vehicles having a communication device, can be useful to set a transition period plan for implanting communication devices onto vehicles and the infrastructure.

The geospatial ontological framework and agent simulation may contribute to increase the intelligence of ITS by supporting data-level and application-level implementation of autonomous vehicle agents to share knowledge in local contexts. This work can be easily extended to support more complex interactions amongst vehicles and the infrastructure.

Acknowledgements

It has been a very long journey, and there have been a lot of twists and turns along the way. This thesis would not have been possible without extraordinary supports from many people.

I would like to express my sincere appreciation to my supervisor, Professor Muki Haklay, for his enthusiasm, guidance and patience. I owe Dr Patrick Weber and Dr Artemis Skarlatidou a real debt of gratitude for their help to untangle the arguments and structure of this thesis throughout the writing process. I want to thank former supervisors, Dr Pragya Agarwal and Dr Rod Bera, who gave me a chance to start this journey. My thanks also go to the examiners, Prof Christopher B. Jones and Dr Claire Ellul. Their thorough comments and advice admittedly helped me to revise the thesis.

Friends and colleagues at CEGE, Chorley and CASA, past and present, have been inspiring and helpful, particularly Amon, Anna, Ateen, Berk, David, Donghan, Gianfranco, Gonzalo, Jenny, Jess, Pinai, Sakpod, Sindy, Sunghyun, Tina, Vyron. I would also like to thank my great family friends in Korea and London for their moral support.

Finally, I am very grateful to my parents, my brother Eun-Kyu, my wife Sookjoo and my daughter Hannah for bearing with this long journey. I love you!

Table of Contents

List of Figures

List of Tables

Glossary

Agent-Based Modelling and Simulation (ABMS)	An Agent-Based Modelling and Simulation (ABMS) is a computational model for simulating the actions and interactions of autonomous agents (individual or collective entities such as organizations or groups) with a view to assessing their effects on the system as a whole. It combines elements of game theory, complex systems, emergence, computational sociology, multi-agent systems, and evolutionary programming.
Beliefs, Desires, and Intentions (BDI)	Beliefs, Desires, and Intentions (BDI) architectures are widely used to reflect agents' goals, and the resulting plans and actions. Beliefs, desires, and intentions are agents' abstract (external) characteristics, and can be transposed into agents' internal characteristics. Beliefs (what they know and what they know how to do) represent an agent's knowledge-base, while desires (what goals they would like to achieve) represent an agent's goals (objectives) and intentions (the goals they are currently committed to achieving) are mapped to plans.
Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE)	The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) is a top-level ontology that has a clear cognitive bias and aims at capturing the ontological categories underlying natural language and human common sense. DOLCE, however, does not commit to a strictly 'referentialist metaphysics' related to the intrinsic nature of the world. Rather, the categories it introduces are thought of as cognitive artefacts, which are ultimately depending on human perception, cultural imprints and social conventions. In this sense, they intend to be just descriptive notions, which assist in making already formed conceptualisations explicit. DOLCE is an ontology of particulars (also known as things), in the sense that its domain of discourse is restricted to them.
Intelligent Transport Systems (ITS)	Intelligent Transport Systems (ITS) are advanced applications which aim to provide innovative services relating to road transport, in which information and communication technologies are applied. It enables infrastructure, vehicles and users to be better informed and make safer, more coordinated, and 'smarter' use of transport networks.
Multi-Agent System (MAS)	A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents within an environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Intelligence may include some methodical, functional, procedural or algorithmic search, find and processing approach.

Ontology	An ontology is 'a model of (some aspect of) the world (domain)' that introduces a vocabulary (of concepts) relevant to that domain, and also specifies the meaning (semantics) of the terms contained in this vocabulary using properties/relations (e.g. is a, is part of) based on a suitable logic.
Resource Description Framework (RDF)	The RDF model is used as a fundamental conceptual description or modelling of information for the Semantic Web. It is similar to classic conceptual modelling approaches such as entity-relationship or class diagrams, as it is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-predicate-object expressions.
Semantic Web	The Semantic Web is an advanced version of the current Web where intelligent software agents can retrieve and manipulate information by encouraging the inclusion of semantic content in Web pages. The Semantic Web aims at converting the current Web of unstructured documents into a Web of data using formal semantic descriptions such as RDF and OWL.
SPARQL Protocol and RDF Query Language (SPARQL)	SPARQL (a recursive acronym) is an official RDF query language, which is a query language for ontology models, able to retrieve and manipulate data stored in RDF or OWL. SPARQL allows four kinds of query form that are SELECT, CONSTRUCT, DESCRIBE, and ASK. SELECT statements return results of matching variables from a query, and CONSTRUCT statements return a RDF graph which is a result of matching variables from a set of triple templates. DESCRIBE forms return an RDF graph to describe resources, and ASK statements return a boolean which is true or false from the matching pattern.
SPARQL Inference Notation (SPIN)	SPIN is a SPARQL-based rule and constraint language that can provide reusable query templates and functions with a SPARQL-friendly syntax. SPIN was designed by combining concepts from object-oriented languages, query languages, and rule-base systems so that it can define a kind of behaviours of ontology classes and instances in the SPARQL-like syntax that can be stored and represented in RDF or OWL. SPIN can define rules and query templates easily using familiar SPARQL expressions. SPIN also provides user-defined functions, which are composed of an argument and a nested SPARQL query.
Tick	Time within an agent simulation environment is regarded as a discrete event whose quantum unit of time is known as a 'tick'. If event `x` and `y` are scheduled at tick `one` and `two` respectively, event `y` will execute after `x`.
Top-level ontology	In information science, a top-level ontology (also known as foundation ontology) is an ontology that describes very general concepts that are the same across all knowledge domains. It supports very broad semantic

interoperability for ontologies under this top-level ontology. It is usually a hierarchy of entities and associated rules (both theorems and regulations) that attempts to describe those general entities that do not belong to a specific domain or application.

Web Ontology Language (OWL)	OWL is another ontological specification for conceptual description or modelling of information of the Semantic Web. It was built on top of RDF and RDF schema to cover their limitations and extend the vocabulary for cardinality constraints, richer property characteristics, etc. OWL can express complex semantics as a vocabulary extension of RDF and RDFS using classes, properties, relations, cardinality, etc.

Abbreviations

ALC	Attributive Language with Complements
ABMS	Agent-Based Modelling and Simulation
ANOVA	Analysis of Variance
BDI	Beliefs, Desires, and Intentions
BFO	Basic Formal Ontology
CA	Cellular Automata
CVIS	Co-operative Vehicles-Infrastructure Systems
DL	Description Logic
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DSRC	Dedicated Short-Range Communication
GIS	Geographic Information Systems
GPS	Global Positioning System
ITN	Integrated Transport Network
ITS	Intelligent Transport Systems
MAS	Multi-Agent Systems
OBE	On-Board Equipment of DSRC
OWL	Web Ontology Language
OWL QL	OWL Query Language
RACER	Reasoner for A-boxes and Concept Expressions Renamed
RDF	Resource Description Framework
RDQL	RDF Data Query Language
RSE	Road-Side Equipment of DSRC
SWRL	Semantic Web Rule Language
SPARQL	SPARQL Protocol and RDF Query Language
SUMO	Suggested Upper Merged Ontology
UML	Unified Modelling Language
URIs	Universal Resource Identifiers

VANET	Vehicular *Ad-hoc* Network
XML	eXtensible Markup Language
W3C	World Wide Web Consortium

1. Introduction

1.1 Challenges in transport systems

Transport is a system that includes activities and a set of facilities consisting of the means and equipment necessary for the movement of passengers or goods from one place to another (Ipfelkofer et al., 2006). There are various transport modes, such as air, rail, road, water, space, etc. Among these transport modes, road transport currently takes the lion's share of the traffic, with 90% of passengers and 82% of lifted goods travelling by road in Great Britain in 2012 and 2010, respectively (Department for Transport, 2013). In addition, it is forecasted that road traffic volume in 2040 will be 43 percent higher than in 2010 (Department for Transport, 2013). This means that, in the long term, the volume of road traffic in the UK will continue to grow. The constant growth of road transport is caused by the combined factors of people, travel, goods lifted, and vehicles on roadways. In this context, safety, mobility, and productivity challenges are significant issues in road transport (Maccubbin et al., 2008).

To address these challenges and improve the performance of the transport system, there are many regulations, such as congestion charges, road tax, pollution tax, and so on. In addition, many technologies have been applied to enhance the performance of road networks. For example, inductive loops in a roadbed, video cameras, and bus detectors at traffic lights have all been used to create efficient traffic flows. Geographic Information Systems (GIS) have also been used within this context.

Geographic information describes features and phenomena on the Earth's surface by using their location in geographic space (x, y, z) and time (Goodchild et al., 1999). It is therefore not surprising that GIS have been applied widely in the area of road transport in order to represent transport activities, as they lend themselves to such descriptions (Goodchild, 2000). A road network and its facilities such as traffic signs, traffic lights, and bus stops can be conceptualised with objects[1] (or geoobjects) in GIS and then can be better managed.

Recently, with the advances made in real-time communications, what were previously just Information Technologies (IT) including GIS, which were applied in the field of road transport,

[1] Objects (or geoobjects) and fields (or geofields) are widely accepted terms for geographic information. They are used to describe discrete entities and continuous processes, respectively (Goodchild et al., 2007). Objects treat identifiable and relevant entities with points, lines, and areas while fields deal with spatial distribution such as elevation, rainfall, and temperature (Worboys and Duckham, 2004).

have started evolving towards Information and Communication Technologies (ICT)[2] under the label of Intelligent Transport Systems (ITS). ITS are advanced applications that aim to provide innovative services and enable various road users to be better informed and make safer use of transport networks (European Parliament and Council of the European Union, 2010). From the perspective of the use of ICT in ITS, this research explores how real-time geographic information can be used to support safety and mobility in an ITS environment in the near future.

In the following section (Section 1.2), the definition of intelligence in the context of this research is presented. The importance of the communication component for advanced transport systems is discussed in Section 1.3. Section 1.4 gives an overview of context modelling and simulation-based approaches for vehicular communications. Then, Section 1.5 addresses the research aim, the research questions, and the research objectives, while the procedural framework and methodology are introduced in Section 1.6. Finally, the structure of this thesis is described in Section 1.7.

1.2 Intelligent infrastructure and vehicles

As this research studies the role of geographic information and communication technologies in an ITS environment, it is important to describe what is meant by an ITS environment. An ITS environment means an advanced form of a road environment, in which each element may play intelligent roles to enhance the road environment itself. Accordingly, this section outlines the intelligence aspect of three elements of road transport.

Intelligence can be defined literally as the capacity to understand semantics in order to reason, make decisions, and solve problems using knowledge and experience (Gottfredson, 1997). Since road transport comprises three elements (i.e. *infrastructure*, *vehicles*, and *operation*), each element has to be considered separately in order to build an intelligent road transport system. Specifically, *infrastructure* refers to fixed structures, such as roadways and road facilities, while *vehicles* are conveyances to move people and lifted goods on roadways. The infrastructure and vehicles are *operated* by humans (e.g. vehicle owners/drivers or members of operating organisations). If the vehicles and infrastructure are intelligent, mutual interaction among vehicles, infrastructure, and humans can occur, which can improve traffic environments. The

[2] Information and Communication Technologies (ICT) have been often used as an extended concept of Information Technology to emphasise on the role of real-time communications (Stevenson et al., 1997).

term ‘intelligent’ for intelligent vehicles and infrastructure can be explained from two aspects: artificial intelligence and ambient intelligence.

On the one hand, intelligent vehicles and infrastructure are instances of artificial intelligence (or intelligent agents) in the road transport domain. An intelligent agent is a flexible and autonomous computational entity that can communicate with other agents in a dynamic and unpredictable environment (Luck et al., 2005). There are many computational entities and sensors which can be referred to as finite state machines or finite state automata. They have input, output, storage, and control units, but they are limited by the fact that their control unit can only change their internal states with already-fixed logics (Linz, 2001). Intelligent agents differ from these computational entities as they are autonomous and have social ability. Intelligent agents can communicate and collaborate with each other to share some information and knowledge, so potentially, they can resolve undetermined situations.

On the other hand, an ITS setting can be seen as an instance of ambient intelligence, since ambient intelligence can be characterised by context-awareness, *ad hoc* networks, and smart sensors (Strang and Linnhoff-Popien, 2004). An ITS setting is composed of intelligent vehicles and roadside facilities which share their traffic situations based on a vehicular network and location sensors such as Global Positioning System (GPS). An ITS setting, as an instance of ambient intelligence, can make the traffic environment responsive, proactive, and sensible by bringing situational information into traffic scenes so that it deals with the needs and desires of intelligent vehicles and the road infrastructure. By connecting vehicles and road facilities, a transport environment can be a rich network in which sensing, computing, and processing powers are embedded.

Therefore, intelligent vehicles and infrastructure can be described as instances of intelligent agents in an ambient intelligence for the road transport domain. At present, drivers gather traffic information mostly from their senses and depend on their audio-visual interpretation of the information. They can also get some general traffic information from the radio or a traffic route map with voice assistance from a navigation system. Traditionally, the infrastructure has been regarded simply as the traffic environment for moving vehicles, which are just carrying people and goods. However, currently, intelligent vehicles and infrastructure already started to assist drivers by acting as intelligent agents with full or partial powers of computing, sensing, and communicating to improve the safety, mobility, and productivity of transport (Maccubbin et al., 2008; Ohmori et al., 2000).

As we have seen so far, intelligent vehicles and infrastructure may provide information and knowledge beyond drivers' visibility so that the efficiency and safety of the whole transport system can be improved. Recently, the automotive industry, including traditional automakers and suppliers and high-tech companies such as Google, is developing sensor-based solutions to enable intelligent vehicles to monitor their surroundings and to identify appropriate navigation paths as well as obstacles and relevant signage in real-time (Silberg et al., 2012; Howard and Dai, 2014). Self-driving car[3] technology represents these sensor-based solutions. As of 2013, four U.S. states (Nevada, Florida, California, and Michigan) and the District of Columbia have passed laws finitely permitting self-driving cars (Smith and Weiner, 2013). In parallel, connectivity-based solutions are also being developed to implement connected vehicles via vehicular communications (Silberg et al., 2012; Howard and Dai, 2014). However, as vehicular communications are based on the responses of neighbouring vehicles and road infrastructure, this is relatively expensive and will take time to implement.

Yet, both solutions and their convergence will be required to 'facilitate adequate mimicking of human senses', and enable intelligent vehicles to compensate for human error and slow reaction times (Silberg et al., 2012). In particular, for intelligent vehicle agents' social ability to share traffic information and knowledge in real-time, the power of communication is essential. The next section outlines how advances in communication technologies influence the applications in transport systems.

1.3 Vehicular communication technologies

When data is processed, organised, structured or presented to answer a specific question in a given context, it is called information. Until recently, the usages of information have been limited to providing services and applications only for human agents (e.g. public usage of public officers and individual usage of individual drivers). However, in an ITS environment, geographic information can assist intelligent vehicles and infrastructure directly. The usage of information (e.g. geographic information, traffic information) can be extended further by supporting the situation awareness of intelligent vehicles and infrastructure.

[3] It is also known as a robotic car, a driverless car, an intelligent car, or an autonomous car.

One approach that can improve the complex interactions of road transport systems is to intervene in transport systems directly by implementing situation-aware vehicles and infrastructure that contain advanced information, communication, and sensing technologies. In this way, vehicles and infrastructure are ‘keeping track of what is going on around’ them, are ‘tightly coupled to the dynamics of the environment’, and are enabled to understand and resolve various traffic situations autonomously as participants in a complex, dynamic transport system (Moray, 2004, p.4).

From this perspective (i.e. intelligent usage of intelligent vehicles and infrastructure), communication and information are tightly coupled to support the interactions of vehicles and roadside facilities. According to the U.S. Intelligent Transport Systems Joint Program Office (Sill et al., 2011), the ITS architecture of the United States can be categorised into four classes: traveller, centre, vehicles, and infrastructure (Figure 1.1). Europe also has a similar ITS architecture with four classes: handheld system, central system, vehicle system, and roadside system (Figure 1.2).

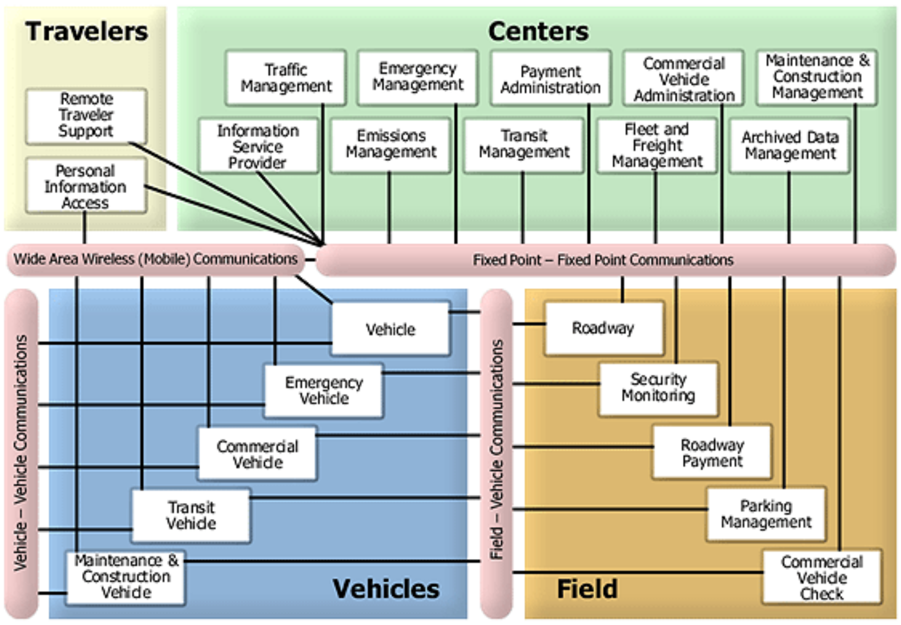

Travelers
Centers
Remote Traveler Support
Personal Information Access
Traffic Management
Emergency Management
Payment Administration
Commercial Vehicle Administration
Maintenance & Construction Management
Information Service Provider
Emissions Management
Transit Management
Fleet and Freight Management
Archived Data Management
Wide Area Wireless (Mobile) Communications
Fixed Point – Fixed Point Communications
Vehicle – Vehicle Communications
Vehicle
Emergency Vehicle
Commercial Vehicle
Transit Vehicle
Maintenance & Construction Vehicle
Vehicles
Field – Vehicle Communications
Roadway
Security Monitoring
Roadway Payment
Parking Management
Commercial Vehicle Check
Field

Figure 1.1 - The national ITS architecture of the United States (taken from www.its.dot.gov/arch/)

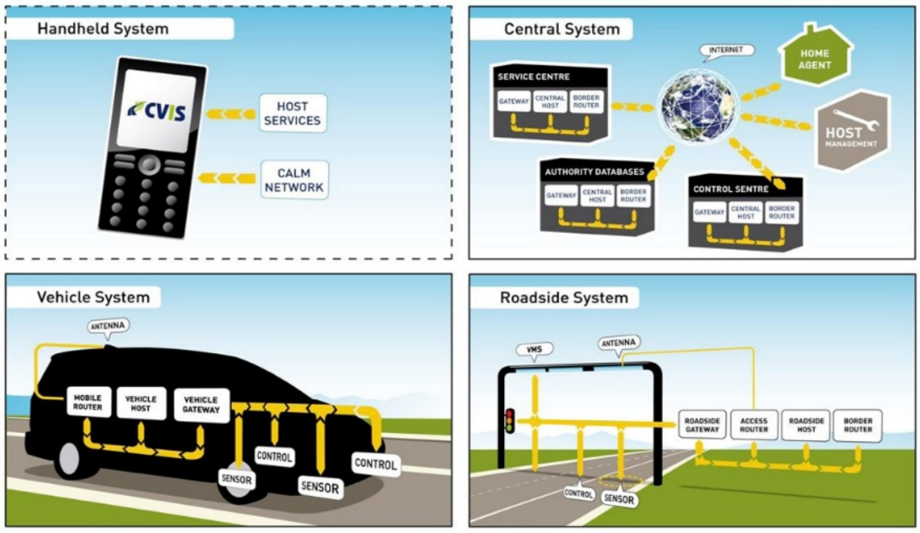


Figure 1.2 - Top-level architecture of Co-operative Vehicles-Infrastructure Systems (Kompfner, 2010)

Theoretically, when complete knowledge of a road network's status is gained, we can find optimal solutions for our purposes (Winter and Nittel, 2006). However, it is impossible to get complete knowledge because a transport system is dynamic and changeable in real time. Therefore, the central communications of transport centres just provide general information to vehicles on a road network. Traffic control centres broadcast car accidents and congestions on main roads using radio and digital signboards, but still, general information is not enough to resolve local traffic situations containing urgent vehicle movements. If there is a local problem that needs to be solved, local *ad hoc* communications among vehicles and infrastructure can be a useful and efficient solution.

As mentioned in Figure 1.1 and Figure 1.2, in an ITS setting, there are four main classes: Vehicle class (VE), Infrastructure class (IN), Centre class (CE), and Traveller class (TR). Among these classes, 16 types of interactions (10 types of two-way interactions) can be provided (Table 1.1) For vehicles (i.e. motorised road users, mainly cars) as the main users of the road transport, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications (shaded parts of Table 1.1) can be of great importance in local situations.

Table 1.1 - Interaction types among four main classes of ITS architecture

Interactions among the four classes	VE_a	IN_a	CE_a	TR_a
VE_b	$VE_a \leftrightarrow VE_b$	$IN_a \rightarrow VE_b$	$CE_a \rightarrow VE_b$	$TR_a \rightarrow VE_b$
IN_b	$VE_a \rightarrow IN_b$	$IN_a \leftrightarrow IN_b$	$CE_a \rightarrow IN_b$	$TR_a \rightarrow IN_b$
CE_b	$VE_a \rightarrow CE_b$	$IN_a \rightarrow CE_b$	$CE_a \leftrightarrow CE_b$	$TR_a \rightarrow CE_b$
TR_b	$VE_a \rightarrow TR_b$	$IN_a \rightarrow TR_b$	$CE_a \rightarrow TR_b$	$TR_a \leftrightarrow TR_b$

There are comfort/entertainment applications, but mostly applications of V2V and V2I communication are related to traffic safety and efficiency, such as intersection collision avoidance, approaching emergency vehicle warning, emergency vehicle signal pre-emption, work zone warning, etc. (Al-Sultan et al., 2014). These Vehicular Ad-hoc Network (VANET) applications can be categorised by communication types and purposes (Table 1.2).

Table 1.2 – Applications of V2V and V2I communications (Gáspár et al., 2014)

Purpose	V2V use cases	V2I use cases
Traffic safety	• Warnings on entering intersections • Hazardous location warning: obstacle discovery, reporting accidents • Sudden stop warnings: forward collision warning, pre-crash sensing or warning • Lane change/keeping warnings/assistance • Privileging ambulances, fire trucks, and police cars	• Warning for hazardous situations (e.g. congestions, accidents, obstacles etc.) • Merging assistance • Intersection safety • Speed management • Rail crossing operations • Priority assignment for emergency vehicles
Traffic efficiency	• Enhanced route guidance and navigation • Intelligent intersections: adaptable traffic lights, automated traffic intersection control, green light optimal speed advisory • Merging assistance: enters an on-ramp to a limited access roadway (it is also a safety application) • Variable speed limits	• Traffic jam notification • Prior recognition of potential traffic jams • Dynamic traffic light control • Dynamic traffic control • Connected navigation

The Dedicated Short Range Communication (DSRC) is a standard that is designed specifically to support short-range V2V and V2I communications between fast moving vehicles and the infrastructure (Fernandes and Nunes, 2007). Of course, other wireless communications such as Long Term Evolution (LTE) can be considered as an additional physical layer to implement an ITS environment, but DSRC is a necessary and not replaceable physical layer of VANET for time-sensitive and safety-critical applications (Mlinarsky and Onishi, 2012; Pinyon Labs/Noblis, Inc., 2010). A more detailed description of DSRC can be found in Appendix A.

This section has demonstrated the importance of communication technologies in an ITS environment. The four main classes of ITS architecture are interconnected with various communication technologies, which allow subsystems of each class to exchange information with each other. Therefore, the use of geographic information in an ITS environment requires also the use of communication technologies, and the contents and data formats of spatiotemporal information can be dependent on communication technologies. The next two sections cover new context models and simulation-based approaches that have originated with the advent of *ad hoc* vehicular communication technology.

1.4 Context modelling and simulation for vehicular communications

Previous DSRC-related research has emphasised the physical capability and capacity of DSRC by focusing on the improvement of the lower layers[4] of the communication (Jiang et al., 2006; Xu et al., 2004; Yin et al., 2004), but the equally important development of suitable and efficient semantic contents for vehicular communications has received notably less attention (Eigner and Lutz, 2008). Recently, in the VANET research community, there are emerging trends of Internet of vehicles (Bonomi et al., 2012; Gerla et al., 2014; Yang et al., 2014) and vehicular cloud computing (Hussain et al., 2012, 2015, Olariu et al., 2011, 2013). With these advanced concepts, vehicles are not only communication resources but also computing resources.

An ITS setting can also be seen as the vehicular version of a ubiquitous computing[5] environment because it is composed of intelligent vehicles and roadside facilities which share

[4] The Open Systems Interconnection (OSI) Reference Model represents the flow of data in a network, from the physical connections (physical layer) up to the user's applications (application layer) (Computer Desktop Encyclopedia, 2011). When two computing devices communicate on a network, the software at each layer on one device assumes it is communicating with the same layer on the other device. The OSI Reference Model includes seven layers: 1) physical layer, 2) data-link layer, 3) network layer, 4) transport layer, 5) session layer, 6) presentation layer, and 7) application layer. The upper layers (Layers 7 through to 4) are more geared to the type of application than the lower layers (Layers 3 through to 1), which are designed to move packets from the sending station to the receiving station.

[5] Ubiquitous computing, which is also called ambient intelligence or pervasive computing, is a trend to make an environment responsive, proactive and sensible by bringing information into our physical world so that it deals with the needs and desires of users (The European Commission's Information Society

their traffic situations based on vehicular communications and location sensors. For the productive use of these resources, vehicles' context-awareness (i.e. context acquisition, context modelling, context reasoning, and context dissemination) is essential to adjust dynamically to the current context (Nassar et al., 2012). A semantic message model is needed for intelligent vehicles to share situational information, understand the context of the messages, and take action in a way that resolves a situation. It is because contexts are mainly related to spatial relations, such as where you are, who you are with, and which object or resource you can use (Dey, 2001). For implementing vehicles' context-awareness, an ontological context model is one relevant candidate that is compatible with *ad hoc* networks and smart sensors.

Strang and Linnhoff-Popien (2004) and Perera et al. (2014) compared existing context modelling techniques (i.e. key-value modelling, mark-up scheme modelling, graphical modelling, object oriented modelling, logic based modelling, and ontology based modelling) at a general level. Then, they evaluated that the ontology-based context model is the most efficient format to manage context, as ontologies 'offer an expressive language to represent the relationships and context', 'share a common understanding of the structure of information among people or software agents'. Ontologies also 'provide comprehensive reasoning mechanisms', 'allow knowledge sharing by decoupling the knowledge from the application and program code', and 'integrate the knowledge on different domains into applications, which are not even known at the design time' (Appendix B).

In the VANET research community, Nassar et al. (2012) preferred ontology-based models for the context representation and exchange between vehicles. Ebers et al. (2013) and Nundloll et al. (2011) also adopted ontological approaches to provide effective 'on-the-fly' interoperability by bridging (classifying, matching, and mapping) semantic differences of the VANET protocols. There have been some studies using ontologies for traffic context models that are compatible with vehicular communications (Eigner and Lutz, 2008; Eigner and Mair, 2009; Barrachina et al., 2012; Section 2.4.2.3). However, they have not addressed how an ontology model can support vehicles' processing and reactions through ontology querying and reasoning.

Apart from the context/messaging model of vehicular communication, the performance evaluation of vehicular communications is another crucial part of implementing ITS

Technologies Advisory Group, 2005). The Ubiquitous computing paradigm can be characterised by context-awareness, *ad hoc* networks, and smart sensors (Strang and Linnhoff-Popien, 2004). Through the connecting of objects, our environment can be a rich network in which sensing, computing, and processing power are embedded (Sharpe and Hodgson, 2006).

environments. Simulation-based approaches have been used because they take advantage of field operational tests (i.e. high degree of realism) and analytical evaluation (closed analytic description for broad conclusions) while avoiding potential dangers/costs of field operational tests and oversimplification of analytical evaluation (Dressler and Sommer, 2015). There are various levels of vehicular mobility modelling in traffic simulation, but in VANET simulation, each vehicle can be seen as a communication subject and a vehicle node at the same time. Therefore, the mobility model of a VANET simulation is designed to provide a general mobility pattern of vehicular nodes rather than a traffic analysis (Härri et al., 2007; Khairnar and Pradhan, 2010), and the level of detail of the mobility model depends on the purpose of the simulation (Section 3.4).

1.5 Research aim, research questions, and objectives

To support vehicles' interaction in local situations, this research adopts DSRC as the physical foundation of vehicular communication. To fill the gap of knowledge in ontology querying and reasoning in such an environment, this research aims to propose a geosemantic ontology model representing communication contents and examine whether it supports vehicles' action/reaction process in local situations. In order to narrow the aspects of the ontology model and its examination, four partial questions were formulated, and the research objectives pursued in order to answer the questions, which are as listed below:

- *Which area of ITS applications will benefit most from intelligent vehicles and their short-range communications? Which traffic situations should be modelled to generate this benefit?*

  There are many well-defined subareas (e.g. traffic management, transportation management, emergency management, vehicle safety systems, etc.) and their subservices already in ITS. Accordingly, there is a need to estimate in which area spatiotemporal information can support vehicles' communications and collaborations better. For a geosemantic message model for intelligent vehicles, this research *explores service areas of ITS applications to find appropriate subservice area, and based on that traffic situations emphasising the benefits of vehicular communications are described.*

- *What are the core contents of the vehicular communications in the traffic situations?*

If there is a local situation and a communicative vehicle (or infrastructure) is responsible for or deeply involved with the situation, the vehicle can generate and send a message to neighbour vehicles for individual and overall safety and efficiency. In this case, the communication message may reflect the perspective of the communication sender (i.e. communicative vehicle or infrastructure). Even though individualised information cannot describe all aspects of a situation as it just delivers partial information about the situation, it can be accepted on condition that the neighbour vehicles can understand what to take immediate action. This research *formulates a set of such descriptions of a situation as communication contents*.

- *How can the contents of the vehicular communications be linked effectively with intelligent vehicles' actions?*

Communication contents can be inputs that provide a basis for intelligent vehicles to understand a situation, so that communication contents can be tightly coupled to the vehicles' internal states (e.g. current location, speed, etc.). Communication contents and vehicles' states and behaviours represent the data layer and the application layer of intelligent vehicles, respectively. To support vehicles' behaviours and decision-making process, this research needs to *develop a vehicle agent model showing the conceptual links between communication contents and vehicles' internal states and behaviours*.

- *If communicative*[6] *vehicles have an influence over the whole system, how can the impact be measured?*

Even though this research deals with communicative vehicles, hybrid situations, whereby communicative vehicles and non-communicative vehicles coexist, have to be considered. This is because these situations are inevitable during the transition towards an ITS environment. With various hybrid situations that have different numbers of intelligent vehicles, the degree of vehicular intelligence and associated benefits can be measured. To provide such situations, this research needs to *develop a simulation platform to examine vehicular communications in various hybrid situations*.

[6] A communicative vehicle is a car that has a vehicular communication device, so that it can communicate with other communicative vehicles and road facilities to send and receive information about a traffic situation.

To answer the research questions and achieve the research objectives, this research used ontological and agent-based approaches for its overall methodological framework. The next section will describe the research procedure and methodology.

1.6 Methodology

As mentioned, the overall aim of this research is 'to propose a geosemantic ontology model representing communication contents and examine whether it supports vehicles' action/reaction process in local situations'. If vehicles (and road facilities) share situational information in advance through *ad hoc* short-range communications, this may facilitate a better understanding and autonomous conflict resolution of various traffic situations. In particular, from the perspective of the convergence of ICT in road transport, the emphasis is on geosemantic contexts and communication contents for vehicles and road facilities.

The research methodology was designed to achieve this research aim by answering the research questions. The three stages of the methodological framework of the thesis, namely, constructing scenarios (Chapter 4), ontology modelling (Chapter 5), and agent simulation (Chapter 6), are shown in Figure 1.3.

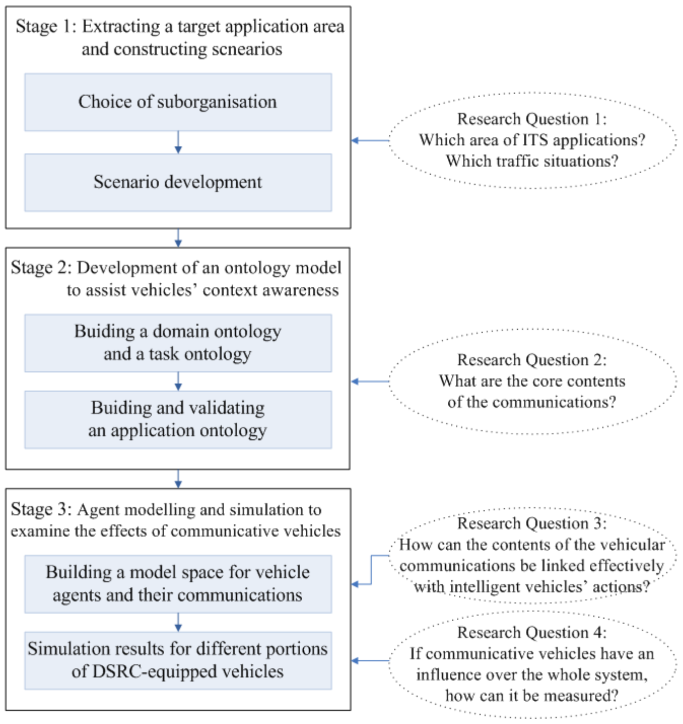

Stage 1: Extracting a target application area and constructing scenearios
Choice of suborganisation
Scenario development
Research Question 1: Which area of ITS applications? Which traffic situations?
Stage 2: Development of an ontology model to assist vehicles' context awareness
Buiding a domain ontology and a task ontology
Buiding and validating an application ontology
Research Question 2: What are the core contents of the communications?
Stage 3: Agent modelling and simulation to examine the effects of communicative vehicles
Building a model space for vehicle agents and their communications
Simulation results for different portions of DSRC-equipped vehicles
Research Question 3: How can the contents of the vehicular communications be linked effectively with intelligent vehicles' actions?
Research Question 4: If communicative vehicles have an influence over the whole system, how can it be measured?

Figure 1.3 - Methodological framework of this research

Stage 1: Selecting a target application area and constructing scenarios that represent the target application area

Generally, an overall system can be seen as multiple suborganisations in agent development, and each agent plays several roles autonomously to achieve the subgoals of the overall system (Zambonelli et al., 2003). So, organisational structures and rules are used to capture and categorise agents' roles/subgoals and their interactions. In this thesis, the organisational concept is adopted to specify the scope of the implementation of this research by choosing a suborganisation, in which short-range communication technology may be useful to resolve a situation. In an emergency, each agent may accept that the most important temporary goal is to assist and resolve the emergency. Therefore, during the emergency situation, each agent's goal can be simplified, and every participant vehicle in the situation can be considered as an agent in a suborganisation. In this way, organisational rules can represent each agent's goals in the situation. A list of ITS user services is used to find a target application area (suborganisation), and after the target service is selected two scenarios are developed to extract basic declarations and rules for the situations. The scenarios show a detailed depiction of reasonable situations, objects and their relations/interactions, which could be used for the development and implementation in the next stages. Detailed descriptions about choosing a target application area and constructing scenarios can be found in Chapter 4.

Stage 2: Development of an ontology model to assist vehicles' context awareness

This research suggests a geospatial ontology to share geosemantic data/information effectively. Generally in top-level ontologies, time-dependent entities are described as events or processes rather than as objects. Moving vehicles in a road network can be modelled as dynamic geoobjects (Goodchild et al., 2007), or dynamic collectives that can also be seen as dual-aspect phenomena that may be presented as either objects or events depending on different viewpoints (Galton, 2005). In this research, vehicles (and road facilities) themselves are seen as objects (i.e. individual elements) as the main bodies of communications and interactions. Consequently, short-range communications and interactions among vehicles (and road facilities) are treated as major events (i.e. the collective actions of groups of individual elements).

Based on several scenarios of communicative vehicles in local situations, an ontology model is developed to support semantic contents for vehicular communications and vehicles' context awareness. Vehicles, a road network, road facilities, and their spatiotemporal relations as described in the scenarios are transformed into the classes and objects of an ontology model, along with their properties and relations. Organisational rules are also expressed as ontology rules in the model. In this way, such a geospatial ontology helps make the vehicles and road infrastructure more context-aware and collaborative. It is fair to say that a geospatial ontology supports a suborganisation if several scenarios of the suborganisation can be redescribed using the ontology model. In an ontology model, any component of a scenario can be expressed in another way using a subset (vehicle instances, road facility instances, and their relations) of the ontology model.

Since this research deals with short-range vehicular communications in an ITS setting, a domain ontology for ITS and a task ontology for vehicular communications were developed. Both ontologies follow the general concepts of the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), which was chosen as the top-level ontology, even though some conceptual amendments are necessary for intelligent vehicle agents, as is further explained in Section 2.4.3. Then, an application ontology is developed to describe the concepts in the scenarios, and some classes of the domain ontology and the task ontology are merged and simplified. To build ontologies, Web Ontology Language (OWL) is used as the ontological language, SPARQL Protocol and RDF Query Language (SPARQL) is used as the query language, and SPARQL Inference Notation (SPIN) is used as the ontology rule language. The application ontology is validated by instantiating vehicles and their semantic relations to fulfil given tasks. Chapter 5 presents the process of the ontology modelling, from developing a domain ontology and a task ontology to developing and validating an application ontology.

Stage 3: Agent modelling and simulation to examine the effects of communicative vehicles

An ontology model developed in the second stage contains classes, objects, and properties (including object properties, otherwise known as binary relations) to describe local emergency situations. It also includes ontology functions and rules that may be useful to resolve the situations. In this stage, an agent-based modelling and simulation framework is implemented to support vehicular communication sharing context information and their spatiotemporal relations to support knowledge sharing among intelligent vehicles and road infrastructure. In this research, the ontology-based messaging model is not tightly integrated to the simulation framework. Instead, a vehicular communication class is provided as a replication of the ontology model to be linked with intelligent vehicle agents' internal states and behaviours, because agents are simulated, not physically implemented.

An agent model is developed to describe the dynamic situations of two scenarios, and then communications and interactions among vehicle agents in local situations are simulated. The agent simulations are performed to check if there are any positive effects of the DSRC-equipped vehicles in several hybrid situations, where each simulation has a different percentage of DSRC-equipped vehicles. In contrast with the fact that altering existing vehicles and infrastructure to provide such situations in a physical way is prohibitively expensive and difficult to achieve, a simulation platform that can provide a virtual experiment environment to test futuristic road situations is preferred.

To build the agent model, some important variables related to vehicle agents in the road environment are presented. Road network geometry and route information are used to build the virtual road environment, and some traffic statistics are used to set individual vehicles' internal states randomly. Action rules for intelligent vehicles in emergency situations are also formulated as vehicles' behaviours. Then, a number of simulations are carried out with different portions of communicative vehicles for each scenario to examine the effects of communicative vehicles in the situations. This research skips the need for the analysis of parameter sensitivity because the simulations will be executed with only one variable parameter, which is the portion of the communicative vehicles. Chapter 6 describes the whole process of the agent modelling and simulation to examine the ontology model in situations of the two scenarios.

1.7 Thesis structure

This chapter introduced the challenges of communication-based spatiotemporal information of ITS in order to resolve local traffic situations. Focusing on the semantic contents of short-range vehicular communication technology, the research questions and the methodological framework have also been presented. To answer the research question and cover the research objectives, the rest of the thesis is structured as follows.

Chapter 2 covers ontology standards and ontological research in related domains. It gives a comprehensive description of ontology and introduces the extant ontology languages to investigate the potential of ontology for sharing information and knowledge. Then, it presents an overview of existing top-level ontologies and three ITS-related domain ontologies.

Chapter 3 reviews the methodologies and platforms used to develop agent systems as well as agent-oriented research in ITS-related domains. From a computer science perspective, it reviews agent-oriented methodologies and agent modelling and simulation platforms for implementing intelligent vehicles and their communications. It also reviews existing agent-oriented research in three different perspectives of ITS-related domains and mobility models for VANET simulations.

In Chapter 4, a target application area is selected from the ITS services, and two motivating scenarios are developed to describe some situations of the target application area. It also outlines spatiotemporal relations and linear referencing to support effective traffic interactions in the situations of the scenarios.

Chapter 5 proposes a geospatial ontological framework with a domain ontology, a task ontology, and an application ontology to support the aforementioned scenarios. For the evaluation of the application ontology, sequential tasks representing the scenarios are instantiated by using the application ontology to show how the ontology can provide support to resolve the situational issues of the scenarios.

Chapter 6 describes agent simulations to provide dynamic environments of the scenarios. Since the spatiotemporal relationships of vehicles and the road infrastructure depend on the location of the vehicles at a particular time, they may change when vehicles move from one place to another. Agent simulations provide dynamic environments, which represent the emergency situations of the scenarios with different parameters. In the simulation environment, the effects of vehicular communications on the traffic flow in an emergency situation are measured.

Chapter 7 provides a general discussion about the research results, namely, the scenarios, the ontology model, and the simulation results. It discusses organisational concepts for the scenario development, ontology modelling for sharing spatiotemporally dynamic situations, and the suitability of the agent simulation for examining the local vehicular communication and its contents.

Finally, Chapter 8 concludes this thesis by reconsidering the research questions and outcomes. The research contributions and the limitations of the research outcomes are discussed, and the suggestions for further research are provided.

2. Ontology in Information Science

2.1 Introduction

This research explores how vehicles and road facilities can collaborate with each other to resolve local situations by acting as intelligent agents. Communication, interaction, and collaboration between intelligent actors thus sit at the core of the research problem. To implement intelligent vehicles and their communications, the vehicles need to have three layers in terms of implementation. The first layer is the physical layer of the vehicular communication, as outlined in Section 1.4, which is outside the scope of this research. Here, the focus will be on the development of an ontology model to describe the spatiotemporal data framework as the data layer of vehicular communication to support local traffic situations, especially for safety-critical applications.

In this chapter, ontological approaches are explored to provide a context/data model for the semantic communication contents of intelligent vehicles. Section 2.2 emphasises the value of an ontological model comparing to other data models. Section 2.3 describes the definition of ontology in information science and reviews ontology standards. Section 2.4 reviews top-level ontologies and ontological research in ITS-related domains. Top-level ontologies and domain ontologies are reviewed to provide a foundation for the modelling basis and semantic possibilities of this research.

2.2 Why ontology?

Section 1.3 dealt with the physical layer of vehicular communication and described how short-range communication standards for the vehicular environment make local communications possible in various traffic situations. As emphasised in Section 1.4, another important facet that facilitates the use of intelligent vehicles and infrastructure is context awareness. Ontology-based modelling is preferred for context modelling in an ITS environment, which is a vehicular version of a ubiquitous computing environment. Vehicular communications provide media for context sharing (i.e. context acquisition, context modelling, context reasoning, and context dissemination). This section particularly draws a comparison between ontological models and other data transfer/storage models.

On the one hand, there are popular data transfer/exchange formats such as Extensible Markup Language (XML) and JavaScript Object Notation (JSON). With a normal programming language, an XML/JSON document could be processed efficiently for an application. However, in this approach, the data exchange format should be application-dependent, and there is no standard for structures. An ITS environment, in which various traffic subsystems and communication protocols exist, may involve many levels of information. Data in XML can be represented in different ways so that an application needs the document type definition (DTD) or the schema to understand the data. However, when there is a need to extend an XML/JSON document that is defined by a DTD or schema, all the applications using the DTD or schema have to be changed, so it is not extensible (Sequeda, 2012). By contrast, in ontology, using triples (subjects-predicates-objects) is the only standardised way to represent data, and a triple already includes schema itself.

On the other hand, there has been a huge change in data storage and management with the advent of the cloud, mobile, social media, and Internet of Things (IoT) applications. There has been a need to develop a new mechanism for storage and retrieval of the non-relational (semi- and unstructured) and hybrid datasets to support a large number of globally distributed concurrent users in real-time and to rapidly adapt to changing requirements. Under these circumstances, leading Internet companies including Google, Amazon, Facebook, Twitter, and Linkedin have established a very flexible database technology called 'NoSQL' to overcome the limitations of the tabular relations used in Relational DataBase Management Systems (RDBMS) and support big data and real-time web applications (McCreary and Kelly, 2013).

There are four types of NoSQL databases, which are key-value store, document store, column store, and graph database. A key-value store (e.g. DyanmoDB, BerkeleyDB) is used for storing simple schema-less data, while a document database (e.g. MongoDB, CouchDB) stores, retrieves, and manages document-oriented data/information in some standard encodings (e.g. XML, JSON). A column store (e.g. Cassandra, BigTable) treats sections of columns of data as data tables to provide high performance in a highly scalable architecture, while a graph database (e.g. Neto4j, InfoGrid) is designed for storing data and their relationships. Social relations, public transport links, and road network topologies can be well represented as graphs.

Triplestore is a type of decentralised directed labelled graph database for the storage and retrieval of triples through semantic queries and ontology-based inference, and also known as Resource Description Framework (RDF) database (see Section 2.3.2 for the details). A triplestore provides a solid standardised query language (performing filtering on values and grouping as well as graph traversal), ontology-based inference (capturing complex logic), and well-defined universal interchange formats, which graph databases are not able to provide (Ontotext, 2014). The central concepts of triplestores have the same foundation with the Semantic Web, which is an advanced version of the current Web where intelligent software agents can retrieve and manipulate information so that computers can deliver the users' requirements more accurately (Berners-Lee et al., 2001).

To sum up, under an ITS environment, data/information can be considered as a partial data/information of the whole situation of the environment, and a distributed and extensible data model is necessary to serve as a messaging model as well as a context model. The triple-based graph representation of an ontology model has potential to deal with various situations, which are beyond the intended design, by providing flexible descriptions, querying, and reasoning.

2.3 Ontology standards

In the transport domain, a data/message model for vehicular communication should be simple and small in terms of the communication device and information transmission. It has to express relations among vehicles and the road infrastructure to describe a situation. In addition, messages transmitted from different vehicles and road infrastructure can be aggregated to describe a shared situation. Since participating vehicles are regarded as the nodes of a vehicular network, the data/message model has to be compatible with a vehicular network as well. In short, the data/message model for vehicular communication needs to express attribute-value pairs and structural relationships, but also be compatible with a network structure.

The transition of the Web environment towards Semantic Web can be a significant reference in terms of the data model and network structure even though it is relatively static compared to the vehicular environment. Under the concept of the Semantic Web, current Web documents will be converted into a web of data/resource in a more descriptive and structured form. Ontologies for the Semantic Web are good examples of such data models providing expressiveness and simplicity as well as network compatibility and scalability (Bergman, 2009).

Therefore, this research adopts an ontological approach both for context modelling and semantic contents for vehicular communications to deal with communication-based spatiotemporal information in an ITS setting. Since an ontology may describe an abstract model or shared knowledge with a machine-readable format, it can express spatiotemporal contexts containing vehicles on the road and traffic events, and their spatiotemporal relations. It can also provide reasoning power based on the open-world assumption so that it may be useful to handle partial information and knowledge of traffic situations.

To develop an ontology, it is necessary to establish how information and knowledge can be modelled in a machine readable format. This section presents the definition of ontology and how it can be used for describing concepts in a format. In the following sections, the definition and history of ontology are presented. Then, the characteristics and potentials of extant ontology languages, ontology query languages, and ontology rule languages for sharing and extracting information and knowledge in a domain are explored. Finally, as a conclusion of this survey for ontology, specific ontology language and ontology query language are chosen for this research to handle situational information in an ITS setting.

2.3.1 Definition of ontology

The ancient Greeks were concerned with the nature of being, existence, or reality as a way of understanding the world or some part of it. An ontology is a catalogue of kinds of being and their relations of dependency as well as their conditions of existence (Marshall, 1998). Ontology was initiated as an explicit part of philosophy and is the study of all things which exist and of categories of things that exist or may exist (Sowa, 2000). On the other hand, the ontology of a certain domain is about vocabularies, terms, and concepts in that domain, as well as their classification, taxonomy, relationships, and domain axioms (Gasevic et al., 2006).

The term 'ontology' was taken from philosophy and began being used by artificial intelligence researchers in the 1980s. For terminological clarification of the term 'ontology', Guarino and Giaretta (1995) proposed using 'Ontology' with an upper-case 'O' to refer to a branch of philosophy and a theory of the nature of being or existence, and using 'ontology' with a lower-case 'o' to represent a logical theory relating to the conceptualisation used by researchers in computer and information science and its various domains.

In computer and information science, an ontology is 'an explicit specification of a conceptualisation' (Gruber, 1993, p.199). Gruber's definition of an ontology is the most quoted and reused by ontology researchers (Table 2.1). Studer et al. (1998, p.184) merged Gruber's and Borst's definition and proposed a definition of an ontology as 'a formal, explicit specification of a shared conceptualisation' of a domain (Gruber, 1993; Borst, 1997; Gómez-Pérez et al., 2004). Recently, Horrocks (2008, p.61) defined an ontology as 'a model of (some aspect of) the world (domain)' that introduces a vocabulary (of concepts) relevant to a domain and specifies the meaning (semantics) of the terms contained in this vocabulary using properties/relations (e.g. is a, is part of) based on a suitable logic (Baader *et al.*, 2009; Horrocks, 2009).

Table 2.1 - Definitions of an ontology (after Gómez-Pérez et al**., 2004)**

Definition of an ontology	Defined By
An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary.	Neches et al., 1991
An ontology is an explicit specification of a conceptualisation.	Gruber, 1993
An ontology is a logical theory which gives an explicit, partial account of a conceptualisation.	Guarino & Giaretta, 1995
An ontology provides the means for describing explicitly the conceptualisation behind the knowledge represented in a knowledge base.	Bernaras et al., 1996
Ontologies are defined as a formal specification of a shared conceptualisation.	Borst, 1997
An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base.	Swartout et al., 1997
An ontology is a set of logical axioms designed to account for the intended meaning of a vocabulary.	Guarino, 1998
An ontology is a formal, explicit specification of a shared conceptualisation.	Studer et al., 1998
An ontology may take a variety of forms, but it will necessarily include a vocabulary of terms and some specification of their meaning.	Jasper and Uschold, 1999
An ontology as 'a model of (some aspect of) the world (domain)' that introduces a vocabulary (of concepts) relevant to a domain and specifies the meaning (semantics) of the terms based on a suitable logic	Horrocks, 2008 Baader et al., 2009 Horrocks, 2009

For concepts, their relations, and constraints in a domain to be accepted by a group or a community that includes users and computers, an ontology has to be in a machine-interpretable format that is explicitly defined by a specific abstract model of shared and consensual knowledge and formal representation (Studer et al., 1998). Therefore, an ontology can refer to and identify a certain phenomenon, topic, or subject area in the world through the representation of classes, properties, axioms, and instances, as highlighted in the following list (Agarwal, 2005; Genesereth and Nilsson, 1987; Gruber, 2009).

- **Class**: A class defines a group of objects that have the same properties. Classes can have hierarchical relations as a super class and subclasses. Terms or words can be used for the name of the class, since they can represent a concept for the class, e.g. *road, traffic light, traffic sign*.
- **Property**: Properties can describe the status or characteristics of a class as well as the relationships between classes. A binary relation between two classes represents their interactions, e.g. *speed limit of a road*.
- **Axiom**: Axioms are defined as restrictions or characteristics of properties which specify the characteristics of classes either partially or fully. So, an axiom is always true in a domain, e.g. *each traffic light controls vehicles on a road*.
- **Instance**: Instances refer to the objects of a domain that are individualised from classes, e.g. *Gower Street*.

These four components represent concepts, their hierarchy, and their relations as well as the semantics of concepts and relations. In a physical space like a road network, they can be used to describe a traffic situation, which includes vehicles, the road itself, and the road's facilities. A traffic situation can be expressed with reference to the four components above. Each element can be generated by sharing situational information from each vehicle's perspective via vehicular communications. To share an ontology, the ontology and its components should be defined and formalised into a computer file. To find a proper way to store and process an ontology for describing traffic situations, we turn in the next section to explore ontology languages.

2.3.2 Ontology languages

As mentioned in the previous section, an ontology has to be formal, so the ontology specification must be 'given in a language that comes with a formal syntax and semantics' in order to make the ontology 'machine executable and machine interpretable' (Staab and Studer, 2009, p.viii). With the four components of ontology (class, property, axiom, and instance), semantics in a domain can be encoded in an ontology language based on a syntactical form of a logic.

There are several ontology languages, such as Knowledge Interchange Format (KIF), Resource Description Framework (RDF), and Web Ontology Language (OWL) (Table 2.2). These ontology languages can be categorised by their base logic, which is either first-order logic or description logic. The detailed description about knowledge representation of first-order logic and description logic can be found in Appendix B.1. Compared to first-order logic, description logic has a low level of complexity, and its concept is similar to object-oriented modelling. In addition, an ontology language based on description logic provides well-defined semantics with formal properties as well as querying and reasoning algorithms (Horrocks, 2010). An ontology language based on such description logic can be a more suitable and convenient basis for knowledge representation of vehicular communications in the transport domain. The ontology model of this research should also be able to describe vehicles' movements, relations, and decision-making procedure efficiently in order to support their intelligence. Therefore, this section focuses on ontology languages based on description logic.

Table 2.2 - Two categories of ontology languages based on its base logic

Structure	Ontology language	Description
first-order logic (FOL)	Knowledge Interchange Format (KIF)	KIF was developed for the interchange of knowledge among different computer programs using declarative semantics.
description logic (DL)	Resource Description Framework (RDF) and RDF Schema (RDFS)	RDF allows to exchange machine-understandable descriptions of resources on the Web using triple patterns. Resources and properties are defined in RDF Schema.
	Ontology Language (OWL)	OWL can be used to make an application which can process Web contents by itself. It provides various vocabulary and formal semantics to develop self-directed Web contents.

In particular, RDF and OWL are reviewed in the following subsections; they are formal ontology languages of the Semantic Web for providing a flexible data model that is suited for specifying a schema and executing queries on an *ad hoc* basis and defining the meaning of data within the context of its interrelationships with other data. Even though Extensible Mark-up Language (XML) and XML Schema can be used for representing, storing, and querying data, it is fair to say that XML is designed for data serialisation rather than informational content. In this research, RDF and OWL can be applied to provide semantics for vehicular communications.

2.3.2.1 Resource Description Framework (RDF) and RDF Schema (RDFS)

The RDF model is based on a simple graph to express triple relations using a triple statement consisting of a subject, a predicate, and an object (Hayes, 2004). To make a triple statement (subject-predicate-object), Universal Resource Identifiers[7] (URIs), blank nodes, and literals can be used. However, a blank node can be a subject in a statement only after it has been mentioned as an object (e.g. *_:a* of Table 2.3), while a literal, such as a string or a number, can only be an object. In Table 2.3, this thesis is described using triples as an example, mentioning the author and publisher of this thesis. The first statement of this example comprised two URIs and one string, and it means that this thesis was created by me.

Table 2.3 - An example of triples

Subject	Predicate	Object
URIref | nodeID	URIref	URIref | nodeID | literal
<http://www.my/thesis/>	<http://purl.org/dc/elements/1.1/creator>	"Seong K. Choi"
<http://www.my/thesis/>	<http://purl.org/dc/elements/1.1/publisher>	_:a
_:a	<http://purl.org/dc/elements/1.1/title>	"UCL"
_:a	<http://purl.org/dc/elements/1.1/source>	<http://www.ucl.ac.uk/>

RDF is a flexible and extendable format to express information using triples (subject-predicate-object) so that it can integrate distributed information which is expressed with various types (Klyne and Carroll, 2004). A set of RDF triples forms a RDF graph, and RDFS semantically extends RDF as RDF's vocabulary description language to specify classes of resources and their properties, such as `rdfs:Datatype`, `rdfs:range`, `rdfs:domain`, `rdfs:subClassOf`, etc. (Brickley and Guha, 2004). Even though RDF represents the binary relations between two resources (a subject and an object) and RDF Schema (RDFS) can express hierarchy and inheritance for classes (`rdfs:subClassOf`) and properties (`rdfs:subPropertyOf`), there are still limitations to expressing detailed semantics such as equivalence between concepts, uniqueness and cardinality of properties, range restrictions, transitive property, etc. (Cardoso, 2006).

[7] The following definition of URIs was written by Connolly (2006) on http://www.w3.org/Addressing/: "Uniform Resource Identifiers (URIs) are short strings that identify resources in the Web: documents, images, downloadable files, services, electronic mailboxes, and other resources. They make resources available under a variety of naming schemes and access methods such as HTTP, FTP, and Internet mail addressable in the same simple way." URIs may not request an actual Web page, whilst Uniform Resource Locators (URLs) do.

2.3.2.2 Web Ontology Language (OWL)

As mentioned in the previous section, RDF and RDFS provide the basic elements of ontologies that can support only simple semantics for describing classes and properties. OWL was built on top of RDF/RDFS to cover their limitations and extend the vocabulary for cardinality constraints, richer property characteristics, etc. As a vocabulary extension of RDF and RDFS, OWL can express complex semantics by using classes, properties, relations, cardinality, etc. (Table 2.4).

Table 2.4 - Comparison between RDF/RDFS and OWL (after McGuinness and Harmelen, 2004)

Ontology	RDF/RDFS	OWL	
Basic Elements	Resources Properties Classes	Based on the basic elements of RDF; add more vocabulary for describing properties and classes	
Vocabulary	type subClassOf subPropertyOf range domain label comment	disjointWith complementOf sameAs equivalentClass Symmetric Transitive inverseOf allValuesFrom someValuesFrom minCardinality maxCardinality	
Expressivity	More flexible (data relationships can be explored from all angles) More efficient (not linear like a traditional database, not hierarchical like XML)	OWL vocabulary allows systems to express and make sense of first order logic for inferences	
Flavours	RDF/RDFS	• OWL1 Lite • OWL1 DL • OWL1 Full	• OWL2 DL - OWL2 EL - OWL2 QL - OWL2 RL • OWL2 Full

The current version of OWL is OWL 2, and it has three sublanguages (profiles), which are OWL 2 EL, OWL 2 QL, and OWL 2 RL (W3C OWL Working Group, 2009). Each sublanguage has different aspects of expressive power, which means a different structure of the ontologies and reasoning tasks for different application scenarios, like those below (Motik et al., 2009).

- OWL 2 EL is particularly suitable for applications that need very large numbers of ontologies (classes and properties). OWL 2 EL provides expressive power for a large biomedical ontology.
- OWL 2 QL is suitable for applications that have very large instance data and need lots of conjunctive queries using AND, OR, and NOT connectives. This profile is designed to store large data in a standard relational database system, and the database system rewrites an ontology query to an SQL query to get query answering.
- OWL 2 RL is suitable for applications that require scalable reasoning and expressive power at the same time. It is compatible with large instance data in the form of RDF triples, and a subset of OWL 2 can be implemented using rule-based technologies.

OWL 2 RL can be considered as suitable for this research because it allows RDF forms and rule-based technologies. Using OWL 2 RL profile, intelligent vehicles can have relative location information in an RDF form as a declarative data model of triples describing a situation. In addition, rule-based expressions can be shared to support vehicles' action plans in specific situations. These two benefits of OWL 2 RL may represent a declarative approach and a procedural approach respectively, which are necessary to implement intelligent systems.

2.3.3 Ontology query languages

To query and process ontology models such as RDF and OWL, RDF Data Query Language (RDQL), SPARQL Protocol and RDF Query Language (SPARQL) are compared in this section.

RDQL has a similar structure to SQL, so users can easily apply their SQL knowledge to ontology querying. There are several systems using RDQL such as Jena, RDFStore, Sesame, PHP XML Classes, 3Store, RAP-RDF API for PHP, Joseki RDF Server, etc. (Seaborne, 2004). However, RDQL does not support Schema (T-box) query and disjunction (OR) operations, so a query can only obtain instance-related results and cannot have OR operators (Table 2.5). SPARQL was developed to complement and extend RDQL, and it has also been an official recommendation from the World Wide Web Consortium (W3C) since January 2008. In contrast to RDQL, SPARQL also provides a bracelet ({}) for using a block of triple patterns (more than one triple pattern), and a single dot (.) divides the triple patterns in a bracelet. In this way, more complex expression is possible in a triple pattern.

Table 2.5 – Comparison between RDQL and SPARQL (Hutt, 2005)

Advanced features	RDQL	SPARQL
Value comparison and data type support	yes	yes
Generalized path expressions	yes	yes
Closure (results of any query operation are also graphs)	no	yes
Optional values (similar to outer join of SQL)	no	yes
Advanced set operations (union and disjunction of graph pattern)	no	yes

SPARQL allows four kinds of query form, namely, SELECT, CONSTRUCT, DESCRIBE, and ASK (Harris and Seaborne, 2012). The SELECT statements return the results of matching variables from a query, and the CONSTRUCT statements return an RDF graph, which is a result of matching variables from a set of triple templates. The DESCRIBE forms return an RDF graph to describe resources, and the ASK statements return a boolean, which is true or false depending on the matching pattern.

A SPARQL query is composed of a result specification, a dataset definition, and a restriction definition. The result specification specifies the contents of fields from a query result with a SELECT clause. In a SELECT clause, fields for the query result have to be provided as variables. The dataset definition part indicates paths for the graphs with a FROM clause or FROM NAMED clause. The restriction definition part defines matching patterns for resources, literals, and properties in triples with the WHERE clause. The basic pattern of a SPARQL query uses triple patterns that have the same variable in a WHERE clause.

The following example represents a SPARQL query and its result from OWL data. In the data part, there is a class hierarchy. The Commercial_Vehicle class is a subclass of the Vehicle class, and the Vehicle class is a subclass of the Feature class. In addition, the data part includes an instance *taxi1* of the Commercial_Vehicle class, as well as its properties and relations. It has current coordinates (longitude geo:long and latitude geo:lat) and is located on roadElement1. The relation between *taxi1* and roadElement1 was built by an object property SpatialRelations:isLocatedOn. Some information and/or knowledge can be extracted from an OWL file. In this example, the query asked to find cars on roadElement1, and then the result showed the cars' ID and coordinates.

```
OWL data:
<rdf:RDF
    xmlns="http://www.mydomain.com/my03#"
    xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource=
         "http://www.ordnancesurvey.co.uk/…/v0.2/SpatialRelations.owl"/>
  </owl:Ontology>
  <owl:Class rdf:about="#VEhicle">
    <rdfs:subClassOf rdf:resource="http://www.opengis.net/gml/_Feature"/>
  </owl:Class>
  <owl:Class rdf:ID="Commercial_Vehicle">
    <rdfs:subClassOf rdf:resource="#VEhicle"/>
  </owl:Class>
     …
  <Commercial_Vehicle rdf:ID="taxi1">
      <geo:lat>51.41299</geo:lat>
      <geo:long>-0.28135</geo:long>
    <SpatialRelations:isLocatedOn rdf:resource="#roadElement1"/>
  </Commercial_Vehicle>
     …
</rdf:RDF>
SPARQL query:
SELECT ?carID ?lat ?long
WHERE
  {  ?car SpatialRelations:isLocatedOn ?roadElements .
     ?roadElements rdfs:label "roadElement1" .
     ?car rdf:ID ?taxiID .
     ?car geo:lat ?lat .
     ?car geo:long ?long .
  }
Query result:
    carID       lat          long
-----------------------------------------
    "Taxi1"    51.42199     -0.28135
```

As vehicles are moving on the road, there is a need to update information related to vehicles' internal states (e.g. coordinates, speed). SPARQL 1.1 contains an update language, which provides update, create, and remove operations (Gearon et al., 2012). The following example uses the same data as the above example to demonstrate a SPARQL/Update query to update the RDF data. If a car is moving and its location can be acquired from a GPS receiver, the current location of the car can be updated repeatedly with a SPARQL/Update query as shown below.

```
SPARQL UPDATE query:
DELETE { ?s geo:lat 51.41299;
            geo:long -0.28135}
INSERT { ?s geo:lat 51.41298;
            geo:long -0.28134}
WHERE  { ?s rdfs:label "taxi1"}
```

Moreover, as a spatial extension to SPARQL for geographic information, there is a standard called GeoSPARQL to support spatial semantics. GeoSPARQL makes spatial queries possible by combining the spatial indexing and computation with ontologies such as RDF and OWL (Perry and Herring, 2012). It can represent relationships between spatial objects (e.g. `distance`, `disjoint`, `intersects`, `touches`, `overlaps`, `equals`, `within`, `contains`, etc.) in a two-dimensional space. For example, it is possible to find all features (i.e. `?targetPolygon`) that the feature `my:aPolygon` contains, where spatial calculations are based on `my:hasExactGeometry` as shown below.

```
GeoSPARQL query:
PREFIX my: <http://example.org/ApplicationSchema#>
PREFIX geo: <http://www.opengis.net/geosparql#>
PREFIX geof: <http://www.opengis.net/def/geosparql/function/>

SELECT ?targetPolygon
WHERE { my:aPolygon my:hasExactGeometry ?aGeom .
        ?aGeom geo:asWKT ?aWKT .
        ?targetPolygon my:hasExactGeometry ?tGeom .
        ?tGeom geo:asWKT ?tWKT .
        FILTER (geof:sfContains(?aWKT, ?tWKT)
                && !sameTerm(?aGeom, ?tGeom))
}
```

2.3.4 Ontology rule languages

There is a W3C standard for rules interchange, Rules Interchange Format (RIF), but it was created to be an interchange format between rule engines rather than to be a rule language (Kifer and Boley, 2010). For this reason, it is limited to general-purpose rules that can be executed directly over RDF/OWL stores (Polikoff, 2011). Therefore, this section outlines two ontology rule languages, Semantic Web Rule Language (SWRL) and SPARQL Inference Notation (SPIN), which are W3C Member Submissions since 2004 and 2011, respectively.

Firstly, SWRL was proposed as a composite of OWL and Rule Markup Language (RuleML), which is a rule language that shares inference rules and that can be stored in an ontology model (Horrocks et al., 2004). It includes a complex abstract syntax to express Horn-like rules for OWL Lite and OWL DL (http://www.w3.org/Submission/SWRL/swrl.owl). The rules are similar to the if-then format exemplified here: if a condition (antecedent) is true, then a conclusion (consequent) will be true. For example, if `x1` is located on `r1` and `x1` is in front of `x2`, then `x2` is located on `r1`.

```
Horn-like rule
locatedOn(?x1,?r1) ∧ inFrontOf(?x1,?x2) ⇒ locatedOn(?x2,?r1)

Abstract syntax
Implies(Antecedent(locatedOn(I-variable(x1) I-variable(r1))
                   inFrontOf(I-variable(x1) I-variable(x2)))
        Consequent(locatedOn(I-variable(x2) I-variable(r1))))
```

Secondly, SPIN is a SPARQL-based rule and constraint language that can provide reusable query templates with a SPARQL-friendly syntax with additional components over the standard SPARQL facilities and SWRL, as listed in Table 2.6 (Knublauch et al., 2011; Mamadolimova et al., 2011). SPIN was designed by combining concepts from object-oriented languages, query languages, and rule-based systems, so it leverages object-oriented principles such as encapsulation and object behaviour to make it easier for machines and humans to understand, interact and process linked data resources (Knublauch et al., 2011). SPIN was chosen as the ontology rule language for this research because it can be tightly coupled within an ontology model to represent some logics. The logic of a rule can be tested with a SPARQL query and then converted into a SPIN rule or template easily because SPIN is an extension of SPARQL. In addition, in a SPIN rule or template, SPIN functions or magic properties (a.k.a. property functions) can be used to avoid reasoning.

Table 2.6 – Functionality comparison between SWRL and SPIN (Mamadolimova et al., 2011)

Functionality	SWRL	SPIN
Rule chaining	yes	yes
Integration with OWL	yes	yes
Built-in/extensible features	yes	yes
Executable on SPARQL support graph server	no	yes
Class rules and constraints	no	yes
Reusable parameterised SPARQL query templates	no	yes
User-defined functions	no	yes
Update values	no	yes
Supports negation	no	yes
Remove triple	no	yes

While each vehicle's location can be treated as an asserted triple and updated by a SPARQL/UPDATE query (see section 2.3.3), spatiotemporal relations among vehicles and road facilities might be acquired by reasoning as inferred triples. However, when dealing with traffic situations that include dynamically moving vehicles, reasoning is not an option to describe their spatiotemporal relations. This is because most inference engines are restricted only to adding new inferred triples from existing asserted triples and rules, and updating already inferred triples is not allowed. This is called monotonic reasoning, in which facts believed true will not be invalidated. If the asserted triples are static and remain still, this reasoning is reliable.

However, in cases where asserted triples need to be updated often, the monotonic reasoning cannot be a good solution because it has to be reset and inferences be run again in order to keep all inferred triples true. There is a way to perform incremental inferences to keep all inferred triples up to date and true after each change, but it is an acknowledged difficult problem in computer science. It can be efficient only if data changes rarely and if queries are executed often. This research deals with moving vehicles, so their locations are changing all the time, and the above is not the case. In the case of moving vehicles, using SPIN functions or magic properties dynamically can be a better choice than reasoning because they can compute property values dynamically on demand (Knublauch, 2011).

For instance, if there is a function `:getDist` and a magic property `:isInFrontOf` to compute relative locations among vehicles, the function and the magic property have a nested SPARQL query. The examples below show the body of `:getDist` and `:isInFrontOf` that have logics to find `:isInFrontOf` relations by calculating their distance using `:getDist` to the coming junction if they are on the same road and moving in the same direction.

```
spin:body of the function :getDist
SELECT ?distance
WHERE {
    ?arg1 :posx ?x1 .
    ?arg1 :posy ?y1 .
    ?arg2 :posx ?x2 .
    ?arg2 :posy ?y2 .
    BIND (afn:sqrt((?x1 - ?x2) * (?x1 - ?x2)
              + (?y1 - ?y2) * (?y1 - ?y2)) as ?distance) .
}
spin:body of the magic property :isInFrongOf
SELECT ?isInFrontOf
WHERE {
    ?arg1 :isLocatedOn ?road .
    ?arg1 :comingJunction ?junc .
    ?isInFrontOf :isLocatedOn ?road .
    ?isInFrontOf :comingJunction ?junc .
    BIND (:getDist(?arg1, ?junc) as ?thisDist) .
    BIND (:getDist(?isInFrontOf, ?junc) as ?otherDist) .
    FILTER (?thisDist < ?otherDist) .
}
```

In terms of triple statements, a magic property can be used as a predicate while the argument of a magic property represents the left-hand side of the statement (the subject). The right-hand side (the object) of the triple is computed by the nested SPARQL query, which is the body of the magic property (Knublauch, 2009). When the `:isInFrontOf` predicate is used in a query, the spin body above will get the results with different bindings (one for the variable `?arg1` and the other for the variable `?isInFrontOf`). The following examples show three different cases of bindings. In the first query of the example, the nested SPARQL query is executed with the variable `?arg1`, which is left blank, and the other variable `?isInFrontOf`, which is pre-bounded with `Vehicle1`. The second query shows the case where the argument `?arg1` is pre-bounded with `Ambulance1`, and `?isInFrontOf` is left blank. Both sides of the predicate can be blank, and the results will return all existing `:isInFrontOf` relations in the whole model.

```
SPARQL query using a magic property with blank subject:
SELECT *
WHERE {
 ?ambulance :isInFrontOf :Vehicle1 .
}

SPARQL query using a magic property with blank object:
SELECT *
WHERE {
 :Ambulance1 :isInFrontOf ?vehicle .
}

SPARQL query using a magic property with blank subject and blank object:
SELECT *
WHERE {
 ?ambulance :isInFrontOf ?vehicle .
}
```

2.4. Ontological research in related domains

The previous section discussed how to describe information in an ontology and how to extract further information by reasoning throughout ontology languages and ontology query languages. RDF and OWL are commonly accepted ontology languages while SPARQL and SPARQL/Update are the latest ontology query languages and provide additional functions.

An ontology helps to describe shared information and knowledge formally and precisely with logical constants and a set of statements. In an ITS setting, an ontology can depict a traffic situation that involves vehicles and road facilities, so it can be used by intelligent vehicles and infrastructure not only to identify one another but also to interact with each other. Consequently, it can be used by intelligent vehicles and infrastructure to be more responsive and to increase the safety and mobility of the transport system.

The semantic contents of communication (i.e. vehicles' current location and the current road status) expressed in an ontology model (OWL) may be updated in real time using SPARQL and SPIN. In terms of semantics and interoperability, an ontology can be useful for shared and consensual knowledge of traffic situations so that it can be a fundamental base for contents of interactions among intelligent geoobjects.

According to Guarino (1998), ontologies can be classified into three levels in accordance with their domain dependence: *top-level ontologies* are domain independent while *domain ontologies*, *task ontologies*, and *application ontologies* are domain dependent.

- *Top-level ontologies* describe general concepts of the world as a whole and relations between the ontologies that are independent across all domains, such as space, time, matter, objects, events, actions, etc.
- *Domain ontologies* and *task ontologies* describe the vocabulary related to a specific domain or a specific task or activity respectively, with the particular meaning of the terms introduced in the top-level ontology.
- *Application ontologies* describe concepts that rely on a particular domain and task at the same time by specializing in related ontologies. There is a close connection between the *application ontologies* and the *roles* played by domain entities for a particular activity.

This section examines existing domain-independent ontologies and domain-dependent ontologies. Even though top-level ontologies are domain-dependent and there are no direct relations between top-level ontologies and ITS-related ontologies, top-level ontologies are important because they deal with a part of the theoretical foundation for all domains. Therefore, in the following sections, top-level ontologies are surveyed before reviewing ITS-related domain ontologies. After that, ITS-related ontological research is reviewed from three different perspectives.

2.4.1 Top-level ontologies

A top-level ontology provides a framework of common knowledge and broad concepts that are general and universal, so domain-dependant ontologies can be composed by importing or extending from a top-level ontology (Semy et al., 2004). If a domain ontology adopts a top-level ontology, the domain ontology can be built based on the theoretical structure and the semantic richness of the top-level ontology.

As the foundation of the domain, the top-level ontology cannot only provide a modelling basis and design patterns, but also can affect the semantic possibilities of the domain ontology with its relevant concepts and logics (Oberle et al., 2007; Semy et al., 2004). Since there are several top-level ontologies, it is necessary to identify which top-level ontology is suitable for the domain ontology of the research.

To find a suitable top-level ontology, Table 2.7 reviews four well-known top-level ontologies, which are Cyc ontology, Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), Suggested Upper Merged Ontology (SUMO), and Basic Formal Ontology (BFO).

Table 2.7 – Four different top-level ontologies

Top-level ontology	Description
Cyc ontology	• A comprehensive knowledge-base to support AI applications from common-sense background knowledge (Matuszek et al., 2006) • Provides context reasoning and problem-solving for various domains, and it can be divided into upper ontology, middle ontology, and lower ontology • The upper Cyc ontology (i.e. top-level Cyc ontology) has three top classes: `Individual`, `PartiallyIntangible`, and `MathmaticalOrComputationalThing` (Cycorp, 2002b)
DOLCE	• The foundational ontologies library for the WonderWeb project by the Laboratory for Applied Ontology (LOA) (Gangemi et al., 2003) • DOLCE is an ontology of `particulars` (i.e. `things`) based on the fundamental distinction between `endurants` and `perdurants` • Top level categories in DOLCE can be split into four classes: `Endurant` (spatial thing), `Perdurant` (temporal thing), `Quality` (entity that is observed or measured, such as shape, sound, smell, colour, and length), and `Abstract` (time interval and space region can be subclasses of this class)
SUMO	• A formal language mapping to all the WordNet (http://wordnet.princeton.edu) words, which consist of nouns, verbs, adjectives, and adverbs following different grammatical rules and is widely used by artificial intelligence researchers • It is composed of SUMO itself, the Mid-Level Ontology (MILO), and a broad range of domain areas, such as communication, countries and regions, distributed computing, transportation, viruses, etc. (Pease et al., 2002) • In SUMO, things can be split into two classes, which are the `Abstract` and `Physical` classes. The `Object` and `Process` classes are subclasses of the

	`Physical` class, and they are conceptually similar to the `endurant` and `perdurant` classes of DOLCE, respectively.
BFO	• Developed to support the scientific research domain (e.g. medical domain, geographical domain, disaster relief domain, etc.) • Provides a single framework for three-dimensional (SNAP) and four-dimensional (SPAN) perspectives, which are continuants (i.e. endurants, static/spatial features) and occurants (i.e. perdurants, dynamic/temporal processes) • `Continuant` is a three-dimensional enduring objects, and `occurant` is a four-dimensional event or process that exists only temporarily in between their initial and terminal boundaries (Grenon and Smith, 2004)

2.4.2 Ontologies in ITS-related domains

ITS can be defined as combinations of computers, databases, maps, and sensors for vehicles and infrastructure (Department for Transport, 2004). Traditionally, GIS covers computers, databases, and maps in the above definition while pervasive computing[8] covers wireless computers and sensors. In terms of the convergence of advanced Information and Communication, we focus on specific traffic situations, in which the definitions of the three domains (i.e. GIS, pervasive computing, and ITS) are compatible (Figure 2.1). This section explores ontological approaches in these three domains.

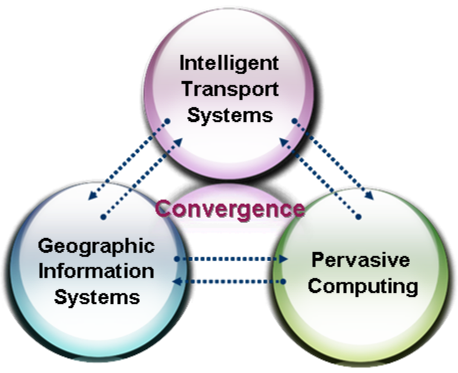


Figure 2.1 - Convergence of some fields based on ICT

The geosensor concept is an example of the convergence of GIS, pervasive computing, and ITS. A geosensor network is a decentralised *ad hoc* wireless network that senses location-related or

[8] Pervasive computing, which is also called ambient intelligence or ubiquitous computing, is a new trend towards ubiquitous appliances that you can access when and where you need it. Hansmann et al. (2003) described pervasive computing through four characteristics, which are decentralization, diversification, connectivity, and simplicity. In Section 1.2, the term ‘ambient intelligence’ was used instead of pervasive computing in order to emphasise the importance of intelligent vehicles and infrastructure.

geospatial phenomena in a ubiquitous computing environment (Nittel et al., 2004a; Winter and Nittel, 2006). Geosensor networks have been used to support dynamic phenomena with monitoring and interaction based on a Mobile *Ad hoc* Network (MANET) in real time (Nittel et al., 2004b). There are studies about *ad hoc* shared-ride trip planning using geosensor networks, which can provide 'instantaneous transportation supply and demand' for local clients who have a mobile device connected to a geosensor network without service pre-notification by hosts or pre-booking by clients (Raubal et al., 2007, p.366). The *ad hoc* shared-ride system was described as follows by Raubal et al. (2007, p.366-367):

> "Advances in technology allow envisioning of an ad hoc, peer-to-peer shared-ride system. This decentralised system will be based on a mobile geosensor network (Winter and Nittel, 2006). In this mobile geosensor network each node is represented by a software agent on a mobile device, either attached to a client or to a host. Clients and hosts have positioning sensors on board, are moving, and can communicate with each other via radio within a limited range. Clients can negotiate directly with nearby hosts for rides."

The above description shows that GIS, pervasive computing, and ITS, are inter-disciplinary and multi-disciplinary technologies. Likewise, ontologies in these three domains can be seen as a convergence in terms of geographic representation and context modelling. Firstly, for the geospatial domain, ontologies can provide a supplemental framework and domain knowledge to represent dynamic geographic information in space and in space-time. Secondly, in pervasive computing, ontologies can provide context models not only for computational entities, such as intelligent agents and sensor networks, but also for various and diverse types of situations. Lastly, ontologies in the transport domain support the geographical representation of vehicles and road infrastructure, traffic contexts, and vehicular communications. Therefore, in the next three sub-sections, a comprehensive review of the relevant work related to ontologies in these three domains is presented from the three different points of view: the geospatial domain, the pervasive computing domain, and the transport domain.

2.4.2.1 Geospatial ontologies

In geographic space, human activities, physical phenomena, and their interactions can be represented as spatiotemporal information with objects, events, and processes. A geographic ontology can be considered as the ontology of geographic space and of the objects and phenomena in geographic space simultaneously (Smith and Mark, 1998). There are two additional points for the ontology of geographic space: not only can it be used to supplement the weaknesses of GIS in representing dynamics phenomena, but it can also function as a formal format for sharing spatiotemporal information and knowledge. Thus, the usage of geospatial ontologies can be classified into three categories: a supplemental framework of dynamic concepts in GIS, a domain ontology of geographic space, and an ontology of the objects and phenomena in geographic space.

Firstly, geospatial ontologies can be used to enrich and expand current GIS concepts. Grenon and Smith (2004) presented a framework for the geographical representation of objects, events, and processes using SNAP-SPAN ontology. There are several research projects designed to expand GIS concepts and functions with SNAP-SPAN ontology because SNAP-SPAN ontology includes dynamic/temporal features in four-dimensional perspectives. Worboys and Hornsby used ontological approaches for geographical objects, events, and processes as well as dynamic representations for moving objects (Worboys and Hornsby, 2004; Worboys, 2005). Goodchild et al. (2007) also proposed a model of dynamic geoobjects through time using three conditions: movement (stationary or moving), shape (elastic or rigid), and internal structure (uniform, evolving).

Secondly, geospatial ontology is a domain ontology for physical space and spatial relations as well as abstract spaces, which can be mapped as features (Reuter and Zipf, 2008). The International Organisation for Standardization's Technical Committee (ISO/TC) 211 provides specifications and standards for geographic information (ISO 19100), and in addition, ontologies for geographic information (ISO 19150) are under development as shown below:

- ISO 19100[9]: Geographic Information/Geomatics (Reference Model/19101, Conceptual Schema Language/19103, Spatial Schema/19107, Temporal Schema/19108, Rules for Application Schema/19109, Methodology for Feature Cataloguing/19110, Spatial Referencing by Coordinates/19111, Spatial Referencing by Geographic Identifier/19112, Metadata/19115, Services/19119, and Geography Markup Language (GML)[10]/19136)

[9] The overview and facesheets of the ISO geographic information series of standards can be found at http://www.isotc211.org/Outreach/Overview/Overview.htm.
[10] GML (http://www.opengeospatial.org/standards/gml) is also a standard of the Open Geosptial Consortium (OGC). GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet.

- ISO 19150[11]: Geographic Information - Ontology (Part 1: Framework/19150-1, Part 2: Rules for developing ontologies in the Web Ontology Language (OWL), Part 3: Semantic operators/19150-3, Part 4: Service ontology/19150-4, Part 5: Domain ontology registry/19150-5, and Part 6: Service ontology registry/19150-6)

For the Semantic Web environment, the W3C Geospatial Incubator Group proposed GeoOWL, a simple ontology model of geospatial resource description, to represent the relationship between a feature and its geometry (Lieberman et al., 2007). As shown in Figure 2.2, a geoobject can be an instance of *gml:_Feature* class which has a *geo:where* property. With a *geo:where* property, the location and the shape of a geoobject can be described with a subclass of the *gml:_Geometry* class such as *gml:Point*, *gml:LineString*, *gml:Polygon*, etc.

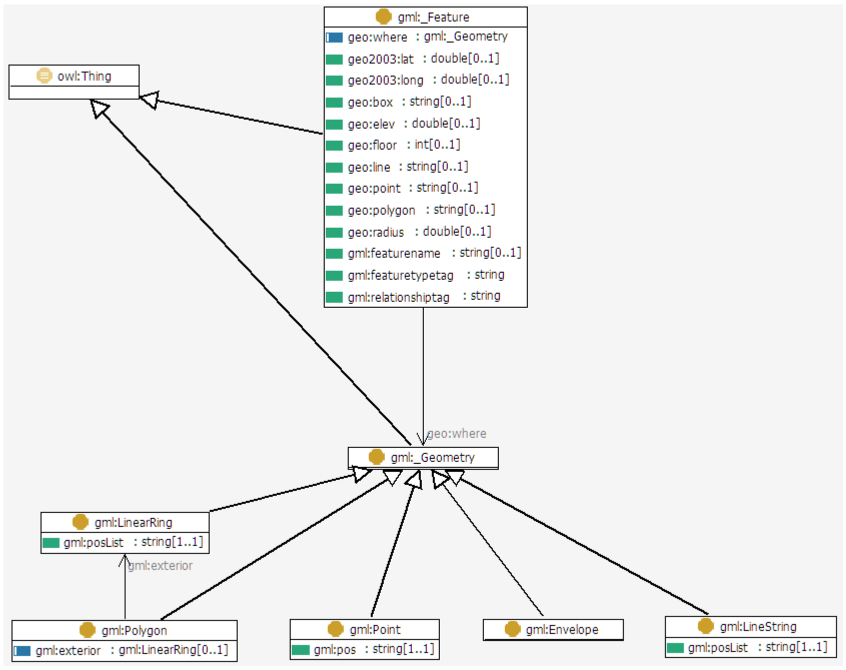
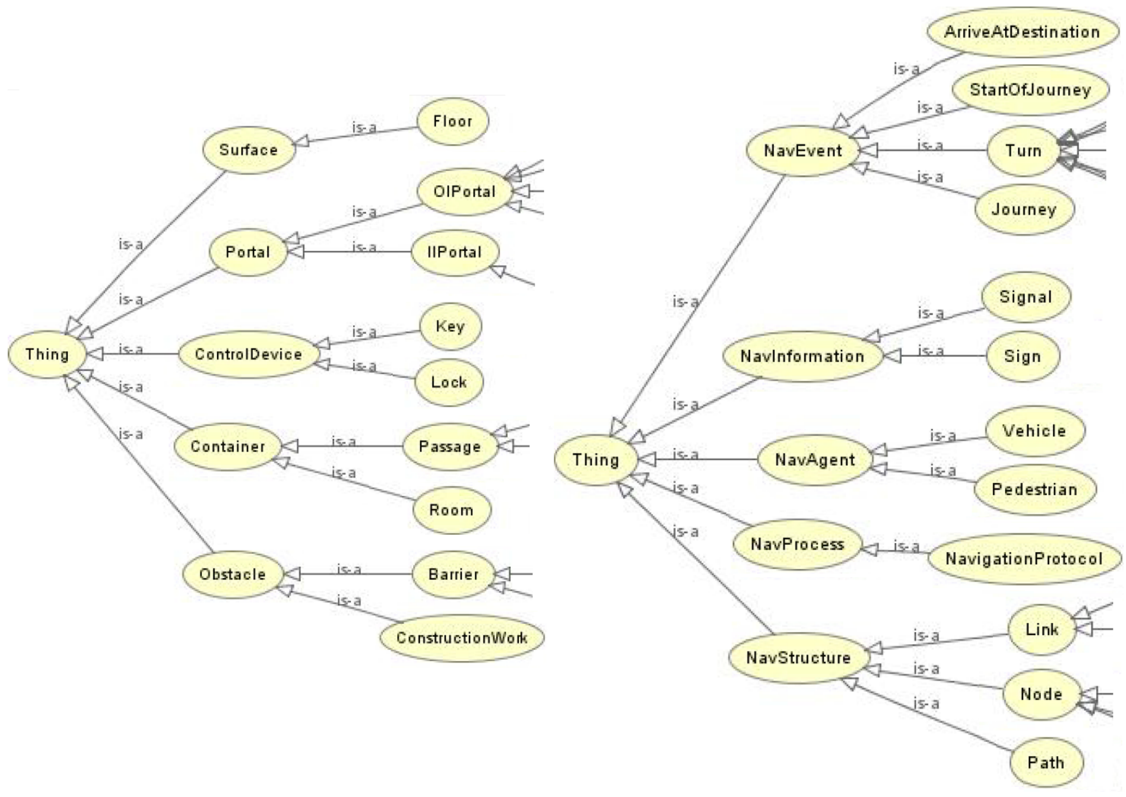

gml:_Feature
geo:where : gml:_Geometry
geo2003:lat : double[0..1]
geo2003:long : double[0..1]
geo:box : string[0..1]
geo:elev : double[0..1]
geo:floor : int[0..1]
geo:line : string[0..1]
geo:point : string[0..1]
geo:polygon : string[0..1]
geo:radius : double[0..1]
gml:featurename : string[0..1]
gml:featuretypetag : string
gml:relationshiptag : string
owl:Thing
geo:where
gml:_Geometry
gml:LinearRing
gml:posList : string[1..1]
gml:exterior
gml:Polygon
gml:exterior : gml:LinearRing[0..1]
gml:Point
gml:pos : string[1..1]
gml:Envelope
gml:LineString
gml:posList : string[1..1]

Figure 2.2 - Classes of GeoOWL

Thirdly, geospatial ontology can be used to describe objects, events, and processes in geographic space. Yang and Worboys (2011) proposed a navigation ontology to provide seamless navigation between indoor space and outdoor space. For pedestrian navigation and vehicle navigation, domain ontologies for indoor space and outdoor space and a task ontology

[11] ISO 19150 was at the end of the enquiry stage of development on 5 June 2014. It was initiated to review the potential and benefit of ontologies and the Semantic Web to achieve the objectives of ISO/TC 211 for the interoperability of geographic information.

for navigation were presented with a well-defined hierarchical taxonomy. For example, the domain ontologies for indoor and outdoor space included concepts such as `Container`, `Passage`, `Connector`, `Surface`, `Obstacle`, `Portal`, etc. (Figure 2.3 a). Rooms and cities can be instances of the `Container` class as they contain something for indoor space and outdoor space, respectively. The `Passage` class is a subclass of the `Container` class, and it can be defined as a way or channel that vehicles or pedestrians can pass through (e.g. outdoor roads, indoor corridors). Outdoor bridges and indoor stairs connect two objects through a barrier, so they belong to the `Connector` class, which is a subclass of the `Passage` class. While the domain ontologies are based on the concept of spaces that contain some physical objects, the navigation task ontology is designed to represent agents' journey on the linear network. The navigation task ontology contains subclasses such as `NavAgent` (i.e. `Vehicle` and `Pedestrian`), `NavStructure` (i.e. `Link`, `Node`, `Path`), and `NavEvents` including various turning events (Figure 2.3 b).



(a) Structure ontology of indoor space (b) Navigation task ontology

Figure 2.3 - A navigation ontology (after Yang and Worboys, 2011)

Using three points of view, this section has shown that geospatial ontologies can be used for the conceptual expansion of geographic representation, the domain knowledge itself, and modelling activities and phenomena in geographic space. The next section will describe how ontologies can support context awareness of intelligent agents in pervasive computing environments.

2.4.2.2 Ontologies in pervasive computing

Intelligent vehicles and an ITS environment can be seen as a specific domain of pervasive computing, however this section reviews ontological approaches in pervasive computing from a broad perspective. An ontology provides a description of the concepts and interrelations for modelling contextual information, making high and formal expressiveness possible as well as reasoning power (Baldauf et al., 2007). Thus, computational agents in a pervasive computing environment, and their goals, plans, actions can be modelled with an ontology to provide contextual information of the pervasive computing environment. Practical and conceptual ontology models are proposed to support various computing entities and sensors of pervasive computing as follows.

Chen et al. (2004) defined a Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA), which has two sets of ontologies - SOUPA Core and SOUPA Extension. SOUPA Core ontologies include concepts like `agent`, `belief-desire-intention (BDI)`, `action`, and `policy`. With SOUPA ontologies, the knowledge, goals, and policies of intelligent agents can be clearly described in the ontologies. SOUPA is a domain ontology for developers of applications in the pervasive computing environment to support knowledge sharing and interoperability so that system developers may focus on the implementation itself rather than building their own ontologies. Furthermore, developers can extend ontologies for their specific purpose by referencing SOUPA Extension ontologies. The Context Broker Architecture (CoBrA) ontology, which was developed for a context-aware system in a smart space, such as an intelligent meeting room, is a good example of an extension from SOUPA (Chen et al., 2004, 2005).

Wang et al. (2004) proposed a context ontology (CONON) for addressing specific issues in context modelling and reasoning in pervasive computing environments. CONON is divided into an upper ontology and domain-specific ontologies not only to share general classes of basic context, but also to provide reusability and flexibility for specific domain knowledge. The upper ontology defines several classes for describing locations, people, activities, computing entities, and other concepts, while the domain-specific ontologies describe detailed features, which may vary in different domains to bring extensibility.

Seremeti et al. (2009) proposed the activity sphere based on an ontology model that describes a task-based activity relating to available resources to support semantic interoperability in a ubiquitous computing environment. Each resource, such as a user, a device, or a service, has a local ontology itself, and for the specific activity, a sphere ontology can be made by aligning and merging local ontologies related to the activity. If a new device joins the activity, the sphere ontology will be updated by the sphere manager and ontology manager. In this way, an ontology is used not only to represent resources with local ontologies, but also to represent an activity as a sphere ontology.

For emergency management, Galton and Worboys (2011) pointed out that ontological conceptualisation can be useful to handle various and diverse types of information in an emergency situation since information events can be collected by trained officials with well-structured procedures and protocols, crowd-sourced data from the public, and automated monitoring devices from sensor networks. According to their information ontology, an information event is enacted by an information agent, who creates a new information bearer by using an information instrument. The content of an information bearer is the information entity that it carries. Their information ontology is composed of several classes, which follow the continuants and occurrents concept of BFO. Information events are the occurrents or perdurants of BFO, while information agents, information bearers (forms of information), information entities (contexts of information), and information instruments (e.g. pens, keyboards, printers, and photocopiers) endure through time as continuants or endurants. In order to support crowd-sourced data and data from sensor networks, the information ontology was expanded to cover collections of sensor readings and sensor-reading events based on the object aggregate (i.e. collection of continuants) and the process aggregate (i.e. collection of occurrents) of BFO. A sensor, which is an information agent, collects information by sensor readings when there are sensor-reading events. An ontology of sensors and sensor readings (as an extension of information ontology) covers various spatiotemporal events (e.g. a time series of sensor readings, a spatial distribution of sensor readings, and a spatiotemporal distribution of sensor readings).

As we have seen so far, ontologies in pervasive computing described situations and events so that computational entities could recognise a change of situation and adopt their action to that changing situation. It also showed that ontological modelling has potential when dealing with various and diverse types and expressions of information captured by various sources from the general public to sensor networks. An ontology model in pervasive computing can be seen as a domain ontology, a task ontology, and an application ontology without strict classification. Since a pervasive computing environment is composed of implemented services and applications, the level of ontologies is relatively unimportant in this domain. However, regardless of the level of ontology, it was shown how the ontologies in pervasive computing supported context modelling and context awareness. The next section will review how ontologies can support the transport domain and traffic situations.

2.4.2.3 Ontologies in road transport systems

Similar to the geospatial ontology, an ontology in road transport systems can be considered as the ontology of the road transport domain and of the traffic situations in road transport. There are two additional points for the ontology of traffic situations: not only can it be used to describe traffic situations, but it can also support the traffic situations directly as an ontology model that is tightly linked to vehicular communication. Thus, the usage of ontologies in transport systems can be classified into three categories: a domain-level ontology, an ontology describing traffic situations, and an ontology model supporting traffic situations directly via vehicular communication.

Firstly, some researchers have focused on developing a domain ontology or a task ontology. Lorenz et al. (2005) presented an Ontology of Transportation Networks (OTN) as an encoding of Graphic Data Format (GDF), which is a map delivery format and a standard for storing geographical data, especially for ITS applications, such as car navigation systems and location-based services. From the ontological notions of GDF themes, they extended OTN classes with timetables for public transport in the `Road_and_Ferry_Features` class and the `Meteorology` class for weather information. Kuhn (2001) proposed a procedure to derive a task ontology for car navigation by analysing the text of the German traffic code. The traffic codes provided various descriptions about the car navigation including objects and actions relevant to driving, driving instructions containing navigation information, and travel narratives explaining observations and decisions during navigation. After selecting suitable actions and objects from verbs and nouns in the text, a cross-tabulation of actions and objects was made to extract entailment relations between objects and actions, and a hierarchy of action classes described activities for car navigation.

Secondly, there are ontologies to describe the relative location of a vehicle on the road and the situations of a car park or of public transport. Hornsby and King (2008) also adopted an ontology using four different motion relations (`isBehind`, `inFrontOf`, `driveBeside`, and `passBy`) in a road environment, based on the assumption that pairs of geosensors are implanted to collect information for moving vehicles. Each case of motion relations between two vehicles represents conceptually different semantics. For example, if an `EmergencyRoadVehicle` object and an `Automobile` have an `isBehind` relation, the `EmergencyRoadVehicle` object can be inhibited on account of the `Automobile` object. The movement of each vehicle at a specific time is possibly stored in a spatiotemporal database, and the motion relations are extracted from the database using queries by a structured query language (SQL). Lee and Meier (2007) proposed an ontology-based spatial context model (Primary-Context Ontology) with object-based traditional context modelling (Primary-Context Model) for the semantics of context information relating to a car parking system. The Primary-Context Model (PCM) provides a uniform storage model for spatial information, and the Primary-Context Ontology (PCOnt) represents their association so that PCM and PCOnt may deliver the semantic meaning of the information as a smart parking space locater service. Wang et al. (2005) proposed an ontology-based public transport query system for Semantic Web and wireless device-based intelligent agents. Vehicle, route, station, and organisation were the four key classes, and they were established using Protégé and Jena software. To provide its query results, the lowest transfer time was used to calculate the shortest path because the lower the transfer time, the lower the trip fare in a public transport domain.

Lastly, there are some studies using ontologies for a traffic context model that is compatible with vehicular networks. An ontological context model was proposed to support collision avoidance applications based on VANET (Eigner and Lutz, 2008). This vehicle ontology provided the necessary data for collision calculation (vehicle ID, current position, current speed, current acceleration, etc.). Each vehicle has an instance of the vehicle ontology and uses it as a packet format for collision avoidance messages. This research showed that importing additional ontologies and modelling environmental aspects or the street network could be useful to avoid redundant geometric calculation, although the model itself got progressively complex. Another ontology model for a wind-gust warning system was proposed to support context-based addressing (Eigner and Mair, 2009). When a warning is triggered, recipient vehicles analyse the message based on the condition of their locations and routes. It shows how an ontology model can be compatible with a context model and vehicular communications. Meanwhile, to share situational information about the vehicles involved in road accidents, Barrachina et al. (2012) proposed the VEhicular ACcident ONtology (VEACON). By using in-vehicle sensors and the ontology (i.e. accident class, vehicle class, environment class, occupant class), an accident vehicle can build a warning message and send the message to the emergency services to ask help for a rescue, and to nearby vehicles in order to prevent additional accidents. The ontology was specially designed to provide an interoperable data structure for both alert information sharing and accidents' severity prediction via vehicular communications. The aforementioned approaches present ontological solutions that are suitable to build traffic context models and exchange vehicular information. However, they have not addressed how an ontology model can support vehicles' processing and reactions through querying and reasoning.

2.5 Summary and discussion

This research explores local traffic situations and vehicular communications to resolve the situations. As a standard model of a directed graph, an ontology-based model has advantages of structuring information using a graph. On top of that, it provides a straightforward and uniform data model supporting decentralised and dynamic schemas, a powerful standard query language, and standardised and implementation-independent data interchange formats. For an ITS environment, in which various subsystems and communication protocols coexist, it has concrete advantages of data portability and interoperability as well as partial information processing, querying, and reasoning. This chapter reviewed the value of an ontological approach (Section 2.2), ontology standards (Section 2.3) and ontological research in related domains (Section 2.4) to build a data layer for vehicular communications.

First, Section 2.2 compared ontological model to other data transfer models and data storage models. It showed that ontology model provides distributed and extensible standardised universal data exchange model and standardised query languages

Second, Section 2.3 explored ontology-related data formats and standards. To implement the ontology model to describe the shared information and knowledge of intelligent vehicles, OWL is used as an ontology language, SPARQL as an ontology query language, and SPIN as a syntax for SPARQL-based rules and functions. To describe vehicles' movements, interactions, decision-making procedures, an ontology based on description logic provides well-defined semantics with object-oriented properties as well as querying and reasoning algorithms compared to an ontology based on first-order logic. OWL is the formal ontology of the Semantic Web, and it is applied to describe semantics of traffic situations related to vehicular communications in this research. SPARQL supports both A-Box query and T-Box query mechanisms and provides powerful expressions. As the aim of this research is to develop an ontology model to support communicative vehicles on the road, the spatial relations of GeoSPARQL are inappropriate to represent vehicles' interrelations. For example, the direct geometric distance (i.e. Euclidean distance) between two vehicles' absolute coordinates cannot represent their network distance (a.k.a. road distance) properly since they are moving on a road network. Therefore, this research focuses on creating its own method to represent vehicles' interrelations. With SPIN, SPARQL's syntax is extended to describe class rules and functions. Topbraid Composer is used as a modelling environment to develop the ontology model. Ontology modelling is based on triple patterns, which are very useful for vehicles to describe relative location information referring to road elements. By sending and receiving relative information, intelligent vehicles can rebuild and re-sketch a situation from their own perspective so that inputs for the decision-making process for vehicles to resolve the situations can be simplified and minimised.

Third, Section 2.4 reviewed four top-level ontologies (Cyc, DOLCE, SUMO, and BFO) and ontological research in three related domains (the geospatial domain, pervasive computing domain, and transport domain). Section 2.4.1 discussed the existing top-level ontologies and related ontology research in the ITS-related domain in terms of the convergence of information and communication technologies. This research deals with vehicular communications, so the ontology model covers a domain/task ontology describing communicative vehicles and an application ontology for practical communications among vehicles. DOLCE and its four super-classes (`endurant`, `perdurant`, `quality`, and `abstract`) provided a clearer framework to describe intelligent vehicles' movements and communications. An intelligent vehicle can be an instance of the `endurant` class, and its travelling or communication can be described as a `perdurant` (Figure 2.4). In addition, spatiotemporal attributes of vehicles, such as location and speed, can be modelled within the `quality` category while the `abstract` category can be used for mathematical or logical axioms or the measurement basis of the `quality` category (Masolo et al., 2003). Traditionally, vehicles and road infrastructure have been categorised into the `Non-Agentive Physical Object` class. However, in an ITS environment, intelligent vehicles (and infrastructure) can be considered as intelligent agents that have communication power as well as sensing, computing, and processing powers. Therefore, they also need to have characteristics of the `Agentive Physical Object` class.

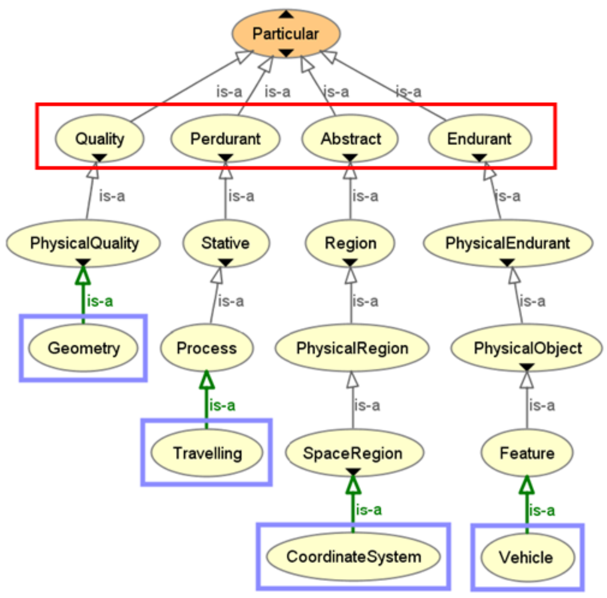


Figure 2.4 - Key concepts derived from the top-level ontology, DOLCE

Even though this research focuses on developing suitable ontologies for vehicular communications, there is a need to describe general geographic and traffic contexts of vehicles and the road environment. Several concepts of the existing domain /task ontologies (e.g. GeoOWL, Yang and Worboys' navigation ontology, OTN) in Section 2.4.2 can be reused to build the domain, task, and application ontologies of this research. For the domain ontology, the feature class of GeoOWL can be referred to describe vehicles' location and road shapes, while the road furniture and road structure classes of OTN can be used to model the road environment. The concept of the subclasses of Yang and Worboys' navigation task ontology will be useful to build the task ontology of this research. In spite of the useful classes and concepts of the existing ontologies, it can be said that the ontologies in the related domain are mostly designed based on existing standards of geographic information or static road environment, so they still have shortcomings in terms of representing vehicles' dynamic processing and reactions via vehicular communications.

To sum up, it is still necessary to create an ontological model for describing intelligent vehicles' movements and communications for this research, which is the intersection part of the three ITS-related domains. To describe intelligent vehicles and infrastructure, some concepts of DOLCE need to be expanded. Similarly, classes of the existing ontologies in the related domains can be adopted or used to develop the domain ontology if the concept of classes is amended to be suitable to an ITS environment. The task and application ontologies for vehicular communications also need to be uniquely created to describe intelligent vehicles' dynamic properties and relations, but still temporal aspects and relative aspects of geographic representations can be referred to conceptually from the existing ontologies.

3. Agent-oriented approach

3.1 Introduction

The previous chapters have described vehicular communication technologies and ontology modelling, which represent the physical layer and the data layer for implementing ITS. For vehicular communication, DSRC was emphasised as the prominent short-range communication technology of ITS for local communications. For data modelling and context modelling, ontology language standards and ontology-related research in the domain were outlined for the semantic contents of vehicular communications.

This chapter explores intelligent agents as the application layer for implementing intelligent vehicles and their communications. Since this research deals with the development of a data layer and an application layer for intelligent vehicles and their communications, the application layer has to be built on top of the data layer, which is the ontological framework. Therefore, an agent-based approach can be used not only to implement communicative vehicles representing the application layer, but also to examine the communication model by empirical validation.

There are two sections in this chapter: agent-based system developments and agent-based research in the related domain. Section 3.2 outlines four capabilities of intelligent agents (autonomous, reactive, proactive, and social characteristics) and the agent-oriented system development process, including the analysis, design, and implementation phases. Agent-based methodologies are reviewed to emphasise new agent abstractions and design/development issues, such as organisational concepts and the environmental model. For the agent implementation phase, agent-based modelling and simulation platforms are reviewed, and one platform is chosen to simulate intelligent vehicles. Section 3.3 reviews agent-oriented research in three domains (GIS, pervasive computing, and transport system) to highlight road transport's geographically distributed, autonomous, and dynamic characteristics. Section 3.4 reviews mobility models that support different macroscopic and microscopic mobility features for different purposes.

3.2 Agent-oriented system development

In the introduction chapter, the definition of intelligent agents was already described focusing on their autonomous and communicative characteristics. To implement intelligent vehicles and road infrastructure, there is a need to understand intelligent agents' essential capabilities and investigate what kinds of agent-based approaches exist and which development platforms could be used. Therefore, this section first describes the four essential capabilities of intelligent agents in more detail, and secondly, reviews agent development processes including 1) methodologies, 2) development platforms, and 3) modelling and simulation platforms (Figure 3.1). After the review of agent-based methodologies, physical development platforms, and modelling and simulation platforms, there is a discussion regarding which methodology and platform is most suited to the purpose of this research.

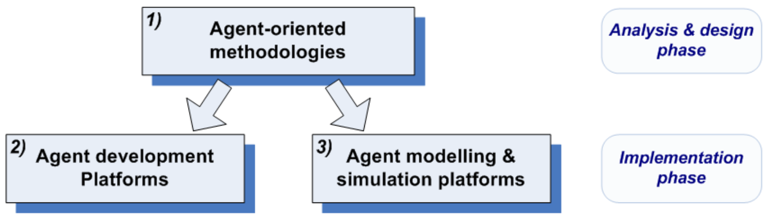

1) Agent-oriented methodologies
2) Agent development Platforms
3) Agent modelling & simulation platforms
Analysis & design phase
Implementation phase

Figure 3.1 - Analysis, design, and implementation phase of agent development process

3.2.1 Capabilities of intelligent agents

In order to develop intelligent vehicles and infrastructure, several characteristics of agent models need to be provided. Wooldridge and Jennings (Wooldridge and Jennings, 1995; Wooldridge, 2009) define four essential attributes of intelligent agents, necessary for flexible autonomous behaviours in different environments, in order to meet an agent model's goals.

- *Autonomous*: Intelligent agents are able to decide for themselves whether or not to perform an action in specific environments in order to achieve their design objectives.
- *Reactive*: Intelligent agents are able to perceive their environment and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.
- *Proactive*: Intelligent agents are able to exhibit goal-directed behaviours by taking the initiative in order to satisfy their design objectives.

- *Social*: Intelligent agents are capable of interacting with other agents (and possibly humans) via *cooperation* (working together as individuals in a team voluntarily to achieve a shared goal), *coordination* (orderly managing/arranging the interdependencies between activities of different groups to maintain unity of action by a central authority), and *negotiation* (solving semantic conflicts among parties and reaching agreements on matters of common interest) in order to satisfy their design objectives.

An ITS environment can be regarded as a dynamic multi-agent environment, and it may have multiple applications running concurrently, implying the ability of intelligent vehicles and road infrastructure (as agents) to multitask. Therefore, an intelligent vehicle must have the autonomy to decide 'what actions it should take at what time' (Zambonelli et al., 2003, p.318), in order to complete its journey by interacting with other vehicles and the road infrastructure. A traffic situation can be described at the aggregate level, but also reflects the decentralised nature of road transport (the individual level). Traffic situations vary from signal change at an intersection to a traffic jam or a traffic accident on a road segment. To deal with dynamic and unpredictable traffic situations, intelligent vehicles need to operate in a flexible way to keep between 'reactive behaviour in response to the environment' and 'proactive behaviour towards the achievement of their designed objectives' (Zambonelli et al., 2003, p.318). In addition, vehicles are moving at high speed on the road, so their coalitions for interactions do not remain static, but constantly change. In this context, agents' social abilities are of special importance. For an agent's social interaction, *coordination* and *negotiation* are as important as *cooperation* to achieve multi-purpose goals. However, this research focuses on the development of short-range communications among vehicles and the infrastructure to resolve local situations. It means that for our purposes, agents' interactions would be mostly treated as their cooperation rather than coordination and negotiation.

To achieve goals, an agent needs plans and actions. The belief-desire-intention (BDI) architecture is widely used to reflect agents' goals and the resulting plans and actions. Beliefs, desires, and intentions are agents' abstract (external) characteristics, and they can be transposed into agents' internal characteristics (Figure 3.2). Specifically, in terms of agent design and implementation, beliefs (what they know and what they know how to do) represent an agent's knowledge-base, while desires (what goals they would like to achieve) represent an agent's goals (objectives) and intentions (the goals they are currently committed to achieving) are mapped to plans. For the agent's actions, each goal has a link to one or more plans.

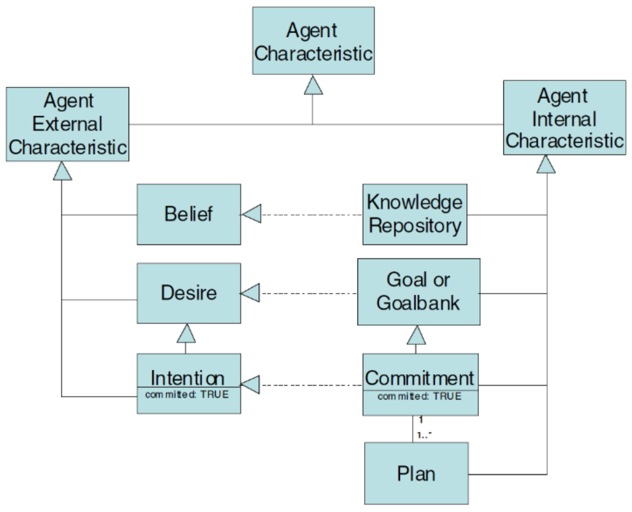


Figure 3.2 - Concepts used in the BDI architecture (Giorgini and Henderson-Sellers, 2005)

3.2.2 Agent-oriented methodologies

The previous section explored agents' characteristics and demonstrated the potential of an agent-based approach to implement intelligent vehicles and infrastructure. When comparing the agent-based approach with the traditional object-oriented view (Booch, 1994), an object is not autonomous and proactive since 'its internal activity can be solicited only by service requests coming from an external thread of control' (Zambonelli et al., 2003, p.321). In addition, an object does not have 'reactive behaviour in response to the environment' because in the environment, it perceives the world only by its internal attributes or other objects (Zambonelli et al., 2003, p.318).

The boundary between the view of objects and agents is getting blurred because of the advent of 'today's distributed and concurrent' characteristics of systems in various domains (Zambonelli et al., 2003, p.318). The Objective C language, for instance, uses Smalltalk-style message passing instead of calling an object instance's method (Apple Inc., 2011), so the sending and receiving of messages look similar to agents' social interaction. However, an object interacts with other objects only when accessing external services and data, so their interactions may show interdependencies between objects, not agents' *social* abilities (*cooperation*, *coordination*, and *negotiation*) as described in Section 3.2.1.

Therefore, for the implementation of agents, new analysis and design methodologies are developed in order to describe new agent abstractions and design/development issues. This section will review the agent-based analysis and design methodologies that describe agents' autonomous abstractions and the environment in which agents will be situated and interact with. There are three methodologies (Gaia, Tropos, and Prometheus) that are most frequently referred to in agent-related research, and the analysis and design phase of these methodologies are summarised in Table 3.1.

Table 3.1 – Analysis and design phases of the Gaia, Tropos, and Prometheus methodologies

Methdology	Analysis and design phase
Gaia methodology	*1. Analysis phase* • Overall system (global organisation) is subdivided into sub-organisations, and an environmental model is made to represent the environment of the suborganisations. • Organisational rules are captured in accordance with the functionalities and competences required by the organisation as well as agents' permissions, responsibilities, interaction patterns. *2. Architectural design phase* • Organisational structure and patterns are created taking into account the organisational rules as well as the organisational efficiency. • Roles and their position in the organisation are captured in a role model. • The topology of the interaction patterns and the control regime of the organisation's activities are defined in an interaction model. *3. Detailed design phase* • An agent model is made, and an agent class of the agent model can be derived from a role or several related roles. • A service model is made to identify agents' interaction with the inputs, outputs, pre-conditions, and post-conditions of each service.
Tropos methdology	1.*Early requirements analysis phase* • The domain stakeholders (humans) as social actors and their dependencies relating to their goals, plans, and resources are identified 2.*Late requirements analysis phase* • Actors are added to the system, along with their functional and non-functional dependencies for the environment. 3.*Architecture design phase* • Subsystems and data/control interconnections are defined as actors and dependencies, respectively. 4.*Detailed design phase* • Agent capabilities, Interactions and the implementation platform are specified..
Prometheus	1. *System specification phase* • Functionalities with inputs (percepts) and output (actions) from the environment are described. *2. Architectural design phase* • Functionalities are assigned to agents, and agents are grouped regarding their interactions when agents share the same data. *3. Detailed design phase* • Agents have a more detailed internal structure to achieve the objectives of targeting an implementing platform.

First, the analysis and design process of the Gaia[12] methodology can be divided into analysis, architectural design, and detailed design phases. This methodology differs from conventional object-oriented methodologies by identifying the importance of the concept of organisational abstractions and environmental models (Wooldridge et al., 2000; Jennings, 2001; and Zambonelli et al., 2003). Agents or individuals can be grouped as an agent organisation or a human organisation by their roles, responsibilities, and interactions. Each agent or individual in the organisation plays one or more roles autonomously to achieve its responsibilities or subgoals for the overall system. Interactions among agents/individuals do not imply their data/control interdependencies, but can be regarded as means (e.g. *cooperation*, *coordination*, and *negotiation*) of realising agents' roles in the system. Therefore, organisational abstractions (organisational rules and organisational structures) offer a realistic perspective to see an overall system as multiple organisations, making possible the capture and appropriate categorisation of an agent's roles/subgoals and their interactions during the analysis and design phases. Figure 3.3 shows an example of an agent system that can be divided into different suborganisations, in which agents can act and interact with other agents and the environment to achieve their different objectives.

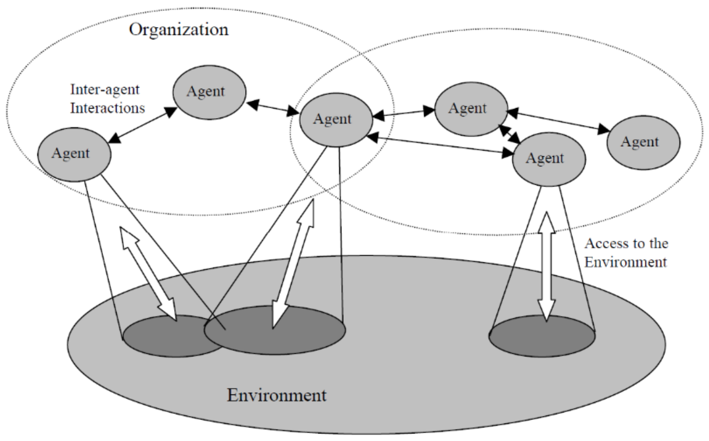


Figure 3.3 - An example of suborganisations (Zambonelli et al**., 2003)**

[12] Gaia means 'the mother Earth' in Greek mythology, and in Gaia theory, all the living organisms on the Earth and the Earth's environment form a single complex system.

Second, the analysis and design process of the Tropos[13] methodology can be divided into four phases: early requirement analysis, late requirement analysis, architecture design, and detailed design. This methodology supports the conceptual modelling of agent-oriented systems with a set of Unified Modelling Language (UML) class diagrams (Bresciani et al., 2001; Bresciani et al., 2004). Throughout analysis and design phases of this methodology, actors, their goals and their plans, as well as their capabilities and their dependencies, are modelled. An actor represents a physical, social, or software agent that has a role or position, and its strategic interests can be described as a goal. There are two kinds of goals, namely, hard goals and soft goals. Hard goals have a clear definition and/or criteria to measure whether they have been achieved or not, while soft goals have not. Actors can execute plans or deliver resources (physical entities or information) to achieve some goals, and their ability to define, choose, and execute a plan in specific conditions or events are described as their capabilities. If there is a dependency between two actors, it indicates that one actor depends on the other to achieve some goals. Various UML diagrams are used to represent these modelling activities graphically: actor diagrams and goal diagrams (for the analysis phase), capability and plan diagrams (for the design phase), and agent interaction diagrams (for the implementation phase). Figure 3.4 shows an example of an actor diagram showing the stakeholders (actors) of an e-culture system. Citizens have a strong intention (hard goal) to get cultural information from the local council since they have paid their taxes, and they expect the taxes will be well spent (soft goal) to provide cultural information and services through the Internet. Figure 3.5 exemplifies an agent interaction diagram to describe interactions between a user and three system agents of the e-cultural system. When the user asks for information, the user interface asks the user for the query specification, and then the user interface queries the directory facilitator to get the address of a free query handler agent that can provide the requested service.

[13] The name of the methodology originates from the Greek word 'trope', which means easily changeable or easily adaptable.

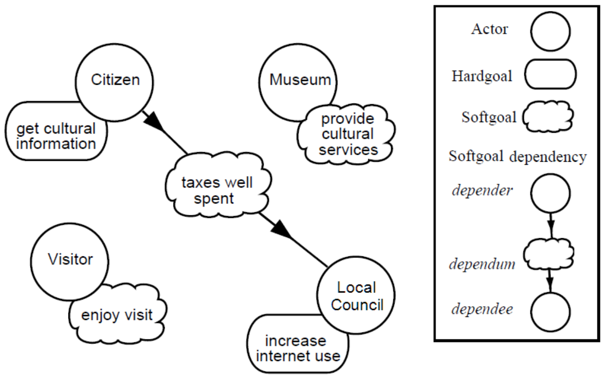


Figure 3.4 – An example of actor diagram model (Bresciani et al.**, 2004)**

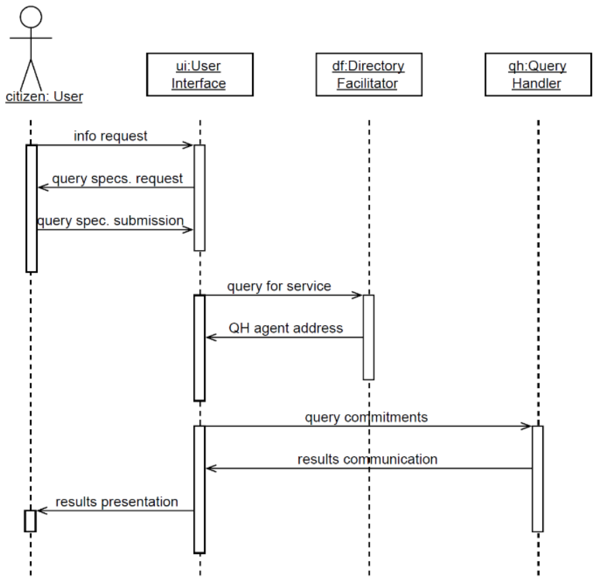


Figure 3.5 - An example of agent interaction diagram (Bresciani et al.**, 2004)**

Third, the analysis and design process of the Prometheus[14] methodology can be divided into system specification, architectural design, and detailed design phases. This methodology draws from and integrates the practical experiences of agent implementations with industry practitioners and undergraduate students (Padgham and Winikoff, 2003). It also uses various diagrams including UML (e.g. interaction diagrams, interaction protocols) throughout the phases.

The analysis and design process of aforementioned methodologies demonstrated not only the concepts (organisational abstraction, the environment model) and notations (agent interaction diagram), but also the processes and techniques of agent-based analysis and design

[14] In Greek mythology, Prometheus was the wisest Titan, and his name means ‘fore-thinker’ because he could foretell the future. He inspired humans by giving useful tools (including fire) to them.

methodologies. These concepts, notations, and processes make possible the development of agent roles (capabilities) and interactions from preliminary models to complete models during the analysis and design process. The next section will review various platforms that can be chosen for the implementation of agents.

3.2.3 Agent-oriented platforms for modelling and simulation

Two methods exist for the implementation of agents: implementing independent agents physically as subsystems that have agent capabilities, or building a virtual environment in a computer in which agents can communicate with each other and perceive the environment. Generally, the former is called Multi-Agent Systems (MAS) while the latter is called Agent-Based Modelling and Simulation (ABMS). Conte et al. (1998, p.3) differentiate between MAS and ABMS by describing the former as 'societies of artificial autonomous agents' and the later as 'artificial societies of autonomous agents', respectively.

There are several Multi-Agent Systems (MAS) platforms (i.e. JASON, JADE, JADEX, and JACK) to implement multiple intelligent agents (i.e. intelligent vehicles and infrastructure) within a physical environment (i.e. an ITS environment). Each platform provides at least two components in order to implement multi-agent systems as shown below (Winikoff, 2005):

- *an agent-oriented programming language* that allows the agent to be written directly using agent concepts (e.g. plans, goals, beliefs) rather than it being encoded in non-agent-oriented languages.
- *a set of libraries or frameworks* providing facilities for inter-agent communication, including facilities for transmitting and receiving messages, and for locating agents.

Apart from that, Agent-Based Modelling and Simulation (ABMS) can be another option to examine intelligent vehicle agents. Experiments based on the physical implementation of intelligent vehicles can be 'costly and restricted to a small number of conditions and repetitions' (Helbing and Balietti, 2012). Therefore, this section focuses on ABMS platforms that can simulate vehicle agents' movements and communications in a virtual road environment on a computer system.

Before the development of ABMS, a social system used to be modelled using a set of rules/equations representing the global behaviours of the aggregate system's individual elements. However, with rules/differential equations, it is almost impossible to model individual interactions, which can be a crucial element of representing a dynamic and complex system. The agent-based simulations have improved the potential of computer simulation as a tool for dealing with social science issues (Conte et al., 1998). In addition, the agent modelling paradigm provides a way to understand the effects of interactions of individuals as a bottom-up approach to the whole system (Crooks, 2010a).

An agent simulation platform provides an artificial environment (space and time) for agents in a computer. A space, as an environment in which agents behave and interact, may be discrete, continuous, or characterised by networks. In general, there are five types of model space: Cellular Automata (CA) model, Euclidean space model, GIS topology model, network topology model, and aspatial (non-spatial) model (Macal and North, 2010). A CA model represents a discrete space while a Euclidean space model and GIS topology model describe continuous spaces. In the CA model, an agent moves from cell to cell on a grid and interacts with its neighbours (4-cell von Neumann neighbourhood or 8-cell Moore neighbourhood). Agents travel in two-dimensional or three-dimensional spaces in the Euclidean space model, while links between agents are defined more generally in the network model. In the GIS model, agents move in a realistic geospatial landscape, and in the aspatial model, the locations of agents are not important and a link between two agents is made randomly.

Time within an agent simulation environment is regarded as a discrete event whose quantum unit of time is known as a 'tick' (Crooks, 2007). If event `x` and event `y` are scheduled at tick `one` and `two` respectively, event `y` will execute after `x`. During the time interval, agents are dynamic in either state (i.e. change) or space (i.e. movement), and they affect other agents' dynamic characteristics (Brown et al., 2005). To represent agents' dynamic characteristics (movements, behaviours and events) in a time step, agent simulation platforms provide scheduled events in three ways (Brown et al., 2005).

- Events may be sequenced in a synchronous step-wise fashion. For example, each agent, set of agents, or non-agent object is signalled to perform its tasks once at each time step or once every n time steps.
- An event may be scheduled to occur only once at some time step *n*. Any number of different events may be scheduled to occur in this fashion providing a predetermined history of events to take place.

- The model may encapsulate 'event-driven' processes whereby model agents may trigger events to occur or may add events to the schedule or queue of events to take place.

The agent-based approach has potential for capturing emergent[15] phenomena, in which individual behaviours can be characterised by if-then rules, or agent interactions, which are heterogeneous and can generate network effects (Bonabeau, 2002). From a social, political, and economic viewpoint, emergent phenomena can be classified into four areas; 1) flows (e.g. evacuation, traffic, etc.), 2) markets (e.g. stock market, strategic simulation, etc.), 3) organisations (e.g. operational risk, organisational design, etc.), and 4) diffusion of innovation and adoption dynamics (Bonabeau, 2002). This research focuses on implementing intelligent vehicle agents that have the communication power to resolve emergency situations on the road, so it can be categorised into the first area of the above categorisation (i.e. traffic flows).

There are several agent modelling and simulation platforms providing not only a framework as a set of standard concepts for designing and describing agent-based models, but also a library of software tools implementing the framework and providing simulation tools (Allan, 2010). The following subsections examine four well-known ABMS platforms: Swarm, NetLogo, REcursive Porous Agent Simulation Toolkit (REPAST), and Multi-Agent Simulation of Neighbourhoods (MASON). These platforms are focused on describing some events and agents' individual interactions, so they are different from agent development platforms (e.g. JASON, JADE, JADEX), instead providing agent concepts (e.g. plans, goals, beliefs) and inter-agent communication specifications. A modeller can simplify and/or abstract events and agents' interaction in his/her own way. Therefore, the following subsections do not indicate the suitability for ontology modelling on each platform.

3.2.3.1 Swarm

Swarm was the first and most mature agent modelling and simulation platform following the 'framework and library' paradigm, which was initially developed at the Santa Fe Institute, USA in 1994, and has been maintained by the non-profit Swarm Development Group

[15] Emergent is the adjective form of emergence. Emergence is the process of complex pattern formation from a multiplicity of relatively simple interactions. Emergence is central to the theories of complex systems.

(www.swarm.org). Originally, Swarm was written in Objective-C, but now Java Swarm allows Swarm's Objective-C libraries to be called from Java so that users can write their part of a Swarm model in Java. In the Swarm platform, a collection of agents is considered as a 'swarm' with a schedule of events over those agents. There are two kinds of swarm: 1) model swarm for the actual model, and 2) observer swarm to observe the model.

3.2.3.2 NetLogo

NetLogo is a multi-platform general purpose complexity modelling and simulation environment from the Centre for Connected Learning and Computer-Based Modelling (CCL), Northwestern University, USA. It is a descendant of StarLogo, which was a modelling environment for K-12 students to explore decentralised systems, and is in use in the social and natural sciences. NetLogo, contains four types of agents: turtles, patches, links, and the observer (CCL, 2011). Turtles are agents moving on patches, which are fixed agents, such as a landscape or a space. A turtle can also have links (relationships) with other turtles to form a network. The observer watches the NetLogo world of turtles and patches providing overall model management.

3.2.3.3 Repast

Repast was initially developed as a Java implementation of Swarm by researchers at the University of Chicago and the Argonne National Laboratory of USA. However, it is a totally independent platform now since it has diverged considerably from Swarm to focus on various social science applications. Repast has been developed for different platforms: Repast J (based on Java language), Repast Py (based on the Python scripting language), and Repast .NET (based on Microsoft.Net framework such as C#). These platforms have now reached maturity and are no longer being maintained. Two new platforms have been released, namely, Repast Simphony in 2007 and Repast for High Performance Computing (Repast HPC) in 2010 (see *http://repast.sourceforge.net/docs.html* for the details).

Repast Simphony supports a new Graphic User Interface (GUI) for visual model development, and a run-time GUI for visual model execution, automated database connectivity, automated output logging, and results visualisation. Also, Repast Simphony has strengthened its GIS support and differentiated itself from other platforms by using Geotools (an open source java

GIS toolkit) and Java Topology Suite (JTS) extensively. For the modelling, there are two new concepts, namely, contexts and projections. A context is a bucket to hold a group of agents, while a projection allows the definition of relationships of agents in a space model (e.g. CA model, Euclidean model, GIS topology model, Network topology model). Agents are added in a context when they are created, and they are removed when they die. A context needs a link with a projection to provide within it a model space for agents.

Repast Simphony provides three programming styles based on Java for the users' different tastes and experiences, as follows.

- *the ReLogo (the Repast dialect of NetLogo) approach* for users with existing NetLogo models, users with limited programming backgrounds, or users want rapid prototyping
- *the flowchart approach* for users who want visual model construction
- *the Java approach* for users who have Java programming experience and want to create highly customised simulations

Repast HPC was designed for users with C++ programming experience and who need to run their models on a large-scale parallel computing platform. Repast HPC also provides two different programming styles: Logo-style C++ approach and standard C++ approach.

3.2.3.4 Mason

Mason is a Java-based open source platform, which was designed as a smaller and faster minimal simulation toolkit for researchers focusing on computationally demanding models by the Evolutionary Computation Laboratory and the Centre for Social Complexity in the George Mason University, USA (Railsback et al., 2006). It is developed especially for researchers who perform many simulation runs, so it was designed to add, remove, and modify features easily (Luke et al., 2004). It also provides some extensions that support GIS and social network topologies so that it can represent agents on the CA model, Euclidean model, network model, and GIS model.

3.2.4 Choosing concepts and platforms for ITS simulation

Section 3.2 reviewed agent methodologies (analysis and design phase), physical agent development platforms, and agent modelling and simulation platforms. After reviewing these methodologies and platforms, this section outlines which concepts and platforms could be chosen and applied to this research. The concept of suborganisations and environmental model in the Gaia methodology is useful to specify the research scope and develop possible ITS scenarios. Also, BDI-related terms and visual notations of Tropos and Prometheus are used to describe intelligent vehicles and their communications. To implement vehicle agents and their communications, the agent modelling and simulation platforms are reviewed for virtual simulation, and Repast is chosen for the agent modelling and simulation platform.

3.2.4.1 Choosing concepts from agent methodologies

By reviewing widely used agent methodologies, it was realised that describing the environment and agents' interactions differentiates agent methodologies from conventional object-oriented methodologies (Section 3.2.2). Based on the organisational concepts (e.g. suborganisations, organisational rules, and organisational structure) and the environmental model, the Gaia methodology provides clear modelling abstractions and techniques to derive agents' role model and interaction model. Meanwhile, Tropos and Prometheus use a similar organisational concept (i.e. actor/subsystem in Tropos, agent grouping in Prometheus) to describe agents' data/control dependencies rather than their interactions and cooperations.

On the other hand, with regard to agent-related terms and notations for modelling, Tropos and Prometheus use BDI-related terms (e.g. beliefs, goals, actions, plans) and visual notations (e.g. goal diagrams, capability and plan diagram, and interaction diagram) during the analysis and design phases in order to specify agent capabilities and interactions. For instance, Tropos and Prometheus use visual notations based on the UML sequence diagram to express agents' interactions (e.g. agent interaction diagram and interaction protocols) while Gaia uses tabular notations (e.g. role schema, protocol definition) to describe roles and interaction protocols.

In conclusion, this research adopts the organisational concepts and the environmental model from the Gaia methodology to define agents' roles and interactions, and uses visual notations of Tropos and Prometheus to express agent classes and their interactions. An ITS setting has well-defined subsystems and applications to support various traffic situations so that using organisational abstractions of Gaia would be very useful to develop vehicle agents in an ITS setting. In addition, vehicles in an ITS setting also interact with their environment via sensors and effectors (see the vehicle to infrastructure interaction in Table 1.1), and the road infrastructure can be designed as a part of the environment or as a set of active agents depending on situations and applications. Therefore, the environmental model of Gaia is also necessary to describe an ITS setting. For the notations for agent classes and their interactions, visual notations of Tropos and Prometheus could be used. In particular, an agent interaction diagram based on the UML sequence diagram could be very practical to describe several interactions among agents with a timeline since the vertical dimension of a sequence diagram represents time.

3.2.4.2 Choosing agent simulation platform

This chapter reviewed four agent modelling and simulation platforms in Section 3.2.3, namely, Swarm, NetLogo, Repast, and Mason. These platforms provide an integrated environment to aid the modelling of agents and their environment as well as to manage running the simulation. Even though each platform has a similar structure for modelling, scheduling, monitoring, and displaying, they still have different characteristics (Table 3.2).

Table 3.2 – Comparison of ABMS platforms (Kravari and Bassiliades, 2015; Zheng et al., 2013)

Criteria	Simulation platform			
	Swarm	NetLogo	Repast	MASON
Terms for Agents	model swarm	turtle	agent, context	model
Terms for Environment	space	patch	projection	field
License	GPL	GPL	New BSD	Academic Free License V3
Open source	yes	no	yes	yes
Simplicity	complicated	simple	simple	complicated
Learnability	average	easy	easy	average
Scalability	average	good	good	average
Performance	average	good	high	good
Stability	average	good	high	good
Robustness	low	average	high	average
Programing language	Objective-C, Java	NetLogo	Java, Python, C++	Java

In terms of usability, scalability and extensibility, Repast Simphony was chosen as the agent simulation platform for this research because it has been developed in a way that applies the virtues of Swarm and NetLogo and provides the most powerful GIS support using Geotools and JTS. In addition, logics and code developed based on Repast Simphony can be reusable when implementing physical vehicle agents using Java-based agent development platforms. Moreover, Repast provides Repast HPC; this is a large-scale simulation platform for high performance computing, but was developed by adopting the same architecture as that used in Repast Simphony. Therefore, to extend the simulation for larger areas or larger datasets, the programming logic and design patterns developed in Repast Simphony can be migrated into Repast HPC easily even though they use different programming languages. While it is beyond the scope of this research, one of the important future research issues is to extend the simulation model with a larger dataset.

3.3 Agent-oriented research in ITS-related domains

As described in Figure 2.1, this research follows the view that GIS, pervasive computing, and ITS domains are tightly linked in terms of the convergence of advanced information and communication technologies. This convergence may be supported by three characteristics of road transport, as summarised by Burmeister et al. (1997): the transport domain is geographically and functionally distributed, subsystems have a high degree of autonomy, and settings are various and dynamic. This section explores agent-oriented research from two different perspectives[16]: the geospatial domain and the road transport domain. Agent-oriented research in these domains demonstrates several ways to simulate agents' interactions in a model space representing a real world geographic environment, to implement intelligent agents to allow the space itself to possess intelligence, and to support dynamic traffic situations with communication technologies, respectively.

[16] Since agent-oriented research in pervasive computing has been mostly focused on indoor applications such as smart homes, classrooms, and workplaces (Cook et al., 2003; Cook et al., 2009; Shen et al., 2005), agent-oriented research in the pervasive computing domain is not included in this section.

3.3.1 Agent-oriented research in the geospatial domain

Agent-based modelling and simulation has various applications in the physical, biological, ecological, social, economic, and management sciences, but in many cases, physical components of real-world systems are modelled as agents and agents' environment (Macal and North 2010). Schelling (1971) modelled economic segregation patterns from unorganised individual behaviours to get collective results, and his model has often been referred to as the first social agent-based model, in which agents and agent interactions represent unorganised individuals and the social process of segregation (i.e. separation, sorting) of their neighbourhoods. However, at that time, there were limitations to developing this kind of model because of a lack of computing power and of any applicable programming mechanism.

Epstein and Axtell (1996) proposed an agent-based social simulation environment called 'Sugarscape' based on a 51 by 51 cellular space, which is considered as the first large-scale agent model. In the Sugarscape space, agents move around to perform various behaviours and tasks, which can vary from simply collecting and consuming sugar to death, reproduction, cultural/informational exchange, combat, trade, disease transmission, and so on. Sugarscape also needs a rule to regenerate its environment; for example, sugar could grow back at different rates in different regions. They introduced various specific scenarios and variables using Sugarscape, so this model has been used as a good introduction to agent modelling and simulation (e.g. NetLogo, MASON).

The majority of agent-based modelling and simulation has been done based on the CA model space because it is simple and easy to apply various rules for agents' states and behaviours. The CA model space has been used for various applications in human geography, such as urban residential segregation (Benenson et al., 2002; Flache and Hegselmann, 2001; Fossett, 2006), urban growth (Batty, 2001), urban sprawl (Rand et al., 2002), the impact of green belts (Brown et al., 2004), and land use/cover change (An et al., 2005; Evans and Kelley, 2004; Evans et al., 2006; Parker and Meretsky, 2004).

Apart from agent-based urban modelling, there has been another trend to model human movement in local areas using GIS data sets (point, line, and polygon features) to represent real-world geographic features for the simulation environment. Haklay et al. (2001) proposed the STREETS model to simulate pedestrian movement in a town centre. In the STREET model, building data were used to create a cellular surface representing agents' walkability, and nodes of a street network, building entrances, and gateway locations (e.g. car parks, on-street parking areas, railway stations, and bus stops) were used for agents' planned routes (i.e. intended sequences of waypoints). To define agents' routes, simple statistical distributions were used to set each agent's socioeconomic group (i.e. shopping preference related to income and gender in this context) and behaviours (e.g. speed, visual range, etc.). In addition, Batty et al. (2003) simulated relatively large numbers of pedestrians for the Notting Hill Carnival in order to predict pedestrian movements in panic situations and evacuation scenarios. A GIS data set (e.g. a road network, the parade route, location points of sound systems, and tube stations) and network analysis of GIS were used to provide a cellular space for agents' accessibility and shortest routes from the parade route and sound systems to tube stations (entry points). For open spaces, Gimblett et al. (2002) applied an agent-based approach to develop a recreational behaviour simulator to support the decision making of environment protection and to evaluate the recreational use of national park environments. Original and alternative trail paths for hikers, bikers, and off-road vehicles of Broken Arrow Canyon, Arizona, USA, are mapped and extracted using a GIS and are used for the simulations.

Even though the aforementioned simulations using CA model spaces showed the potential of agent-based approaches and GIS data sets describing urban phenomena and human behaviours, CA model spaces (a series of discrete cells) still have limitations when presenting geometric details of various geographic features that have different sizes (e.g. buildings and houses) and patterns (e.g. linear features, such as roads) (Crooks, 2010b). Since several simulation platforms have started to support the GIS topology model space as one of their model spaces making it possible to use vector-based GIS data sets (e.g. shapefiles) for the agent environment, it is possible to use a more explicit geographical space in the modelling and simulation process nowadays.

Crooks (2010b) presented an agent-based segregation model using vector GIS data to represent the urban environment explicitly. He applied the Schelling's model to simulate residential segregation within a hypothetical cityscape, and explored how geographic features (e.g. points, lines, and polygons) might be affected during the simulation process. Two vector layers were used: a polygon layer representing the model space of urban environment and a point layer representing agents. The model was based on the Repast platform and Java Topology Suite (JTS), which is a set of Java-based libraries to provide spatial analysis functions. In the simulation, a point agent uses the standard overlay operators (e.g. point-in-polygon, buffering, and intersection) of JTS to calculate its neighbourhood and boundaries of area taking into consideration physical features, such as rivers, motorways, and railways in the urban environment. It showed how geometry and spatial analysis operation can be integrated directly into the simulation process on a continuous space.

Malleson (2010) adopted an agent-based model to represent the process and dynamics behind crime, especially for residential burglary. A vector-based GIS data set of Leeds (e.g. UK administrative boundaries, roads, buildings), acquired from the Ordnance Survey MasterMap Topography layer and the Integrated Transport Network (ITN) layer, was used for the accurate representation of the urban environment (i.e. household-level or street-level environment). In addition, the Leeds crime dataset (e.g. crime locations, date/time of offence, and/or victim information, nominal offender data), deprivation statistics, and Output Area Classification (OAC) indicating the social characteristics of local areas, were used to extract spatiotemporal patterns of city-wide burglary in Leeds. Household-level factors (i.e. accessibility, visibility, occupancy, security, attractiveness, and traffic volume) and community-level factors (i.e. community cohesion, community similarity) are considered to simulate residential burglary risk in Leeds on the Repast Simphony platform.

Manley and Cheng (2011) presented multi-agent simulation of drivers' reactions to replanning their route after an unexpected road closure and subsequent delay across urban road networks. They simulated individual drivers' behaviours by defining agents' personal knowledge and experience of the road network in order to predict the movement of non-recurrent congestion in response to a road closure. It was implemented using the Java-based Repast Simphony platform, and the road network of London was acquired from the Ordnance Survey MasterMap ITN layer. Driver agents are categorised into three profiles for different levels of spatial knowledge: taxi driver, commuter, or tourist. A taxi driver's knowledge covers the whole road network while commuters know partial information, and tourists know only the key roads. Although the simulation used simple abstractions of drivers' behaviours, it predicted the distribution of vehicle flows and the non-recurrent congestions caused by a road closure.

We have reviewed agent-based approaches that model social agents and their unorganised behaviours/interactions in terms of the agents' movements in the geographic space. CA models have been widely used to represent a geographic space, but recently, the use of vector datasets for agent simulation is increasing because the geographic features of a vector dataset can represent a geographic space with several layers that have different dimensions (e.g. points, lines, polygons, and 3D features). Particularly, linear features can provide a better representation of roads in a geographic space to simulate in more detail the dynamic movements of pedestrians or vehicles because their movements in each time 'tick' are not limited and regularised by a cell size. The next section will review in more detail how agent technologies can support the transport domain and traffic situations.

3.3.2 Agent-oriented research in transport systems

Similar to the agent-oriented research in the geospatial domain, agent-based modelling and simulation has been adopted to analyse and design various applications in road transport including traffic management, traffic guidance/control, and capacity/resource management (Burmeister et al., 1997). In spite of the potential of an agent-based approach for various possible applications, this section focuses on reviewing agent modelling and simulation for real-time traffic guidance and control based on vehicles' interactions and communications.

Hallé and Chaib-draa (2005) simulated a platoon of cars to increase highway traffic density and safety. For collaborative driving, they adopted a hierarchical architecture of three layers (guidance layer, management layer, and traffic control layer). Each vehicle has a guidance layer to control the vehicle actuators (longitudinal control and lateral control), while a management layer determines the inter-platoon and intra-platoon communications and a traffic control layer supports V2I communications (e.g. sign boards, traffic signals). It showed that the decentralised platoons was more flexible than the centralised platoons because each agent could make a choice to increase its own safety despite the fact that it exchanged more messages than the centralised model.

Dresner and Stone (2005, 2004) proposed a reservation-based agent system to handle traffic congestion at an intersection. They divided an intersection into reservation tiles (cells), and each tile could be reserved by one car per time stamp. To improve the coordination protocol, seven message types are used. A vehicle sends the intersection a message (`request`, `change-request`, `cancel`, `reservation-completed`) containing its expected arrival time and speed to the intersection, its maximum and minimum acceleration, its length and width, etc. The intersection agent simulates the vehicle's journey and assigns intersection cells to the vehicle by sending a message (`confirmation`, `rejection`, `acknowledgment`) at each time stamp.

Eichler et al. (2005) presented a simulation environment that combined a network simulator and a traffic simulator in order to investigate the benefits of vehicle-to-vehicle communications. Vehicles were regarded as moving nodes of a communication network and moving agents on a road network at the same time. A scenario was used in which a car broke down and sent a warning message to surrounding cars, and the simulation result showed that, in most cases, wireless enabled cars received the warning message from the broken-down car and found a new route to avoid getting stuck in congestion.

Kesting et al. (2008) simulated inter-vehicle communications on freeway traffic to resolve a stop-and-go wave situation by store-carry-forward (SCF) strategy using the opposite driving direction. A communication message is generated by a communicative vehicle on the occasion of a local speed change and sent to a communicative vehicle in the opposite driving direction. The message is then sent back to a communicative vehicle in the original driving direction. The recipient vehicle can predict the future traffic situation. In this way, two communicative vehicles in the same driving direction can communicate with each other beyond the communication range by using a communicative vehicle on the opposite driving direction as a relay station.

In the urban setting, the design of broadcast routing protocol and vehicular cooperation can be more challenging because of complex road geometry and intersections. Viriyasitavat et al. (2010) proposed a new routing protocol for SCF strategy in the urban setting and simulated the performance of the protocol. Desai et al. (2013) investigated an agent-based congestion avoidance solution by performing cooperative route-allocation decisions at junctions along the route. Every vehicle agent exchanges its preferred route, undertaking internal processing of successive virtual negotiations to calculate overall utility and individual utility, and then shares the final resulting allocations. It showed that the local negotiation strategy is a promising strategy for effective traffic route allocation and congestion management.

We have reviewed agent-based modelling and simulations describing specific traffic situations and vehicular communications. The traffic situations represent cars on the motorway, cars at an intersection, and a car breakdown. The simulation results show that the safety and mobility of transport can be improved if vehicles and/or the road infrastructure are situation-aware and autonomous, and the situation awareness of vehicle agents can be acquired by vehicular communications. Therefore, it can be said that vehicular communication is a compulsory component for both agent simulation and physical agent implementation in the transport domain for the transition towards ITS.

3.4 Mobility models in VANET simulations

In the simulation of vehicular communications, it is essential to model the road traffic and movement patterns of mobile vehicles to evaluate the performance of vehicular communication. This section outlines the different usages of the mobility model in traffic simulations and VANET simulations and then provides mobility features in various VANET mobility models.

Road traffic simulations can be classified into three types, which are macroscopic, mesoscopic, and microscopic, depending on the level of detail of the model representing vehicle mobility, traffic flow and road network topology (Yin-fei et al., 2015). A macroscopic traffic model describes traffic flow as a whole continuous flow that can be defined by macroscopic parameters such as speed and density, so it does not draw the details of individual vehicles such as lane change. A mesoscopic traffic model describes inflow and outflow of vehicles on road sections and intersections, so a group of vehicles on a certain road section is treated as a unit to be modelled. A microscopic model deals with each vehicle as a basic unit of the traffic flow so that it can reflect individual vehicles' microscopic behaviour such as car following, overtaking, lane changing, and so forth. Currently, various transport planning authorities, agencies and consultants have adopted simulators to model individual vehicles and their behaviour to evaluate/predict the impact of planned development or changes of a road infrastructure (Papageorgiou and Maimaris, 2012). TRANSIMS, CORSIM, VISSIM, PARAMICS, AIMSUN, SimTraffic, and TransModeler are such traffic simulators supporting macroscopic and microscopic levels of simulations (Henchey et al., 2013; Khairnar and Pradhan, 2010).

Meanwhile, in the VANET research community, vehicles are treated as nodes of a network and determining the regulation of node movements is the essential part of the simulation (Al-Sultan et al., 2014). In addition, when evaluating network routing protocols, it is necessary that a mobility model is integrated with generic network simulators such as OPNET, NS-2, NS-3, and OMNet++ (Pan, 2008). Aforementioned commercial traffic simulators such as TRANSIMS, CORSIM, VISSIM, PARAMICS provide models of exact vehicular mobility and vehicular behaviour for traffic analysis, but they require commercial licenses and do not provide an API for the integration with a network simulator. For these reasons, commercial traffic simulators are not popular in the VANET research community, and open-source vehicular mobility models providing global vehicular mobility patterns for VANET simulation are widely accepted instead. In a VANET simulation, the mobility model is designed to provide a general mobility pattern of vehicular nodes rather than very detailed vehicular movements for traffic analysis (Härri et al., 2007; Khairnar and Pradhan, 2010).

A vehicular mobility model for the VANET simulation can be developed based on synthetic models, survey-based models, trace-based models (Table 3.3); or existing mobility models can also be used. The features of the existing major vehicular mobility models that can import GIS dataset for the road topology are compared in Table 3.4. Vehicles' macroscopic mobility features (i.e. multi-lane, initial and destination position, trip generation, path computation, and velocity) are related to generate an initial traffic flow. The microscopic mobility features (i.e. car following model, lane changing model, intersection management model) have to be taken into consideration to model vehicles' general interaction. Nevertheless, all the microscopic mobility features are not compulsory for the vehicular mobility model, and the level of detail of the mobility model can be simplified depending on the purpose of the simulation.

Table 3.3 – Vehicular mobility models for VANET simulation (Härri et al., 2007; Yin-fei et al., 2015)

Models	Description
Synthetic models	- Mathematical models reflecting a realistic physical effect - Five classification (Fiore, 2006) • *Stochastic models* wrapping all models containing purely random motions • *Traffic stream models* looking at vehicular mobility as hydrodynamic phenomenon • *Car Following Models*, where the behaviour of each driver is modelled according to vehicles ahead • *Queue Models* which models roads as FIFO queues and cars as clients • *Behavioural Models* where each movement is determined by a behavioural rules imposed by social influences for instance
Survey based models	- By including large scale surveys and gathered extensive statistics of real traces into a mobility model, a generic mobility model is able to reproduce the non random behaviour observed in real daily life urban traffic - Surveys and statistics of real traces over commuting time, lunch time, traveling distance, preferred lunch politics
Trace based models	- Extracting generic mobility patterns from movement traces - Extrapolating patterns not observed directly by the traces

Table 3.4 – Mobility features of the vehicular mobility models (Härri et al., 2007; Manikandan and Dhas, 2012; Yin-fei et al., 2015)

Models	Macro-mobility			Micro-mobility			
	Multi lane	Init/Dest position	Velocity	Multi lane	Init/Dest position	Lane changing (overtaking)	Multi lane
RiceM (Saha and Johnson, 2004)	no	r	u	no	no	no	no
MOVE (Karnadi et al., 2007) SUMO/TraNS (Behrisch et al., 2011)	yes	r, ap	s, r-d	yes	yes	no	no
STRAW (Choffnes and Bustamante, 2005)	no	r	s	yes	yes	no	no
GrooveSim/ GrooveNet (Mangharam et al., 2006, 2005)	no	r	u, r-d	no	no	no	yes
SSM/TSM (Mahajan et al., 2006)	no	r	u, r-d	no	yes	no	no
VanetMobiSim (Härri et al., 2006)	no	ap	u, r-d	yes	yes	yes	yes

r: random, ap: attraction point, u:uniform, s: smooth, r-d: road dependent

3.5 Summary and discussion

This chapter reviewed agent development paradigms (Section 3.2), agent related research in related domains (Section 3.3), and mobility models in VANET simulations (Section 3.4). Agent methodologies and various simulation platforms are reviewed in Section 3.2. From the methodologies, organisational concepts (of Gaia) and visual notations (of Tropos/Prometheus) were chosen to design and describe agent's roles and interactions. Organisational concepts are used for categorising a whole ITS domain into subsystems to specify the application area of this research. In addition, visual notations are applied to design intelligent vehicles' interactions and communications. Repast Simphony was chosen as an agent modelling and simulation platform.

Following this decision, Section 3.3 reviewed agent-based approaches in the geospatial domain and transport domain to show the potential of vehicle agents' autonomy and capabilities based on context awareness and social interactions. The geospatial domain and transport domain showed a pattern for use with agent modelling and simulation approaches focusing on agents' movements, behaviours, and social interactions among people or vehicles in the geographical or traffic environment. In the transport domain, some agent-related research has started to simulate vehicles' communications to show the potential of vehicular communications for traffic guidance and control in real-time. Section 3.4 reviewed various mobility models for the VANET simulation. The level of detail of the macroscopic mobility features and microscopic mobility features can be decided depending on the purpose of the simulation.

In this research, an agent simulation is constructed to assess the effects of agent communications in road transport. For the simulation, there is a need to develop a vehicular mobility model to generate a mobility pattern of vehicular nodes. The ontological communication message model, which will be developed here, can be used or replicated for the simulation.

4. Constructing scenarios

4.1 Introduction

In the previous two chapters, ontological approaches and agent-based approaches were reviewed and discussed from the perspective of intelligent vehicle agents. An ontological model can support the development of intelligent vehicle agents as a context model as well as the message contents of their communications. In addition, an agent model can be used to simulate the dynamics in traffic situations related to vehicles' movements and communications in order to measure the effect of vehicular communications, which are based on the ontological framework.

This chapter outlines the first stage of the methodological framework (Figure 1.3), choosing a target application area and specifying possible traffic situations and vehicular communications as scenarios. In Section 4.2, ITS user services are regarded as the whole organisation of ITS applications, and a user service is chosen to describe emergency traffic situations, in which DSRC can provide support to resolve the situations. In Section 4.3, two scenarios are proposed to describe an ambulance situation and another situation of a car breakdown on the motorway. Section 4.4 describes existing traffic interactions and potential traffic interactions based on vehicular communications. A conceptual diagram based on spatiotemporal relations among vehicles and road facilities on a road network is presented, and the relative location representation of vehicles is described to support intuitive communication contents from the perspective of individual vehicle's and to simplify the decision-making process of vehicles.

4.2 Target application area selected from ITS user services

As a multi-agent system is considered as a computational organisation, organisational concepts are increasingly used to analyse and design multi-agent systems as a crucial component of agent-oriented methodologies (Zambonelli et al., 2001). The organisational role, as the main organisational concept, identifies agents' roles within the system and the interaction protocols in which the different roles are involved (Zambonelli et al., 2001). As discussed in Chapter 3, the Gaia methodology provides three additional organisational concepts, namely, organisational rules, organisational structures, and organisational patterns, to support the organisational role model.

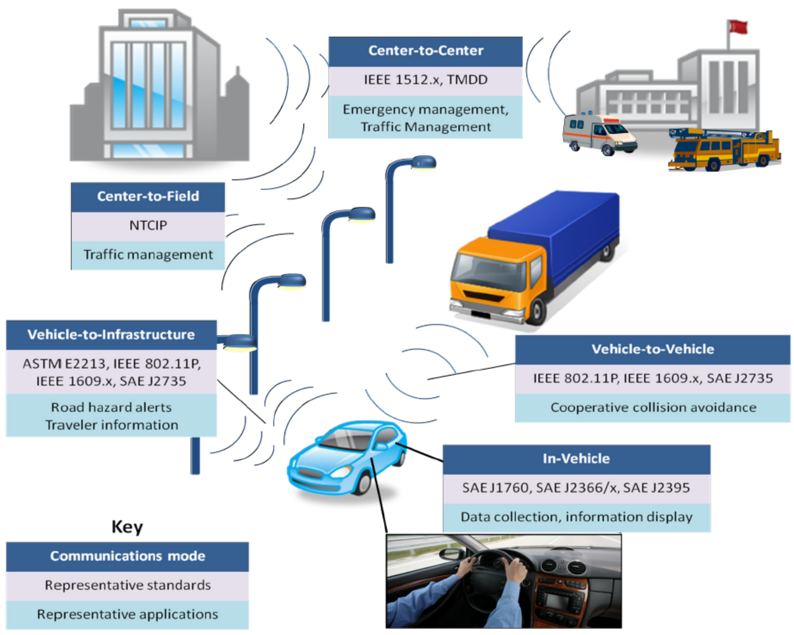

Center-to-Center
IEEE 1512.x, TMDD
Emergency management, Traffic Management
Center-to-Field
NTCIP
Traffic management
Vehicle-to-Infrastructure
ASTM E2213, IEEE 802.11P, IEEE 1609.x, SAE J2735
Road hazard alerts Traveler information
Vehicle-to-Vehicle
IEEE 802.11P, IEEE 1609.x, SAE J2735
Cooperative collision avoidance
In-Vehicle
SAE J1760, SAE J2366/x, SAE J2395
Data collection, information display
Key
Communications mode
Representative standards
Representative applications

Figure 4.1 - The environment, suborganisations, and interaction media of an ITS environment (from Sill et al., 2011)

As shown in Figure 4.1, a multi-agent system can be comprised of several computational suborganisations. These suborganisations are implemented as applications and services that constitute one big computational organisation/system. An ITS environment can be seen as a multi-agent system, ITS applications/services and their communication technologies correspond with the suborganisations and interaction media of a multi-agent system. In this section, ITS services are considered as suborganisations, so one suborganisation is chosen that is related to emergency traffic situations using local vehicular communications. Section 4.2.1 lists the whole organisation of ITS applications, and service bundles supporting a vehicle's situation-aware characteristic by communications are highlighted in Section 4.2.2 and Section 4.2.3. Lastly, the target application area is chosen in Section 4.2.4.

4.2.1 User services as the whole organisation of ITS applications

To choose a target area, ITS user services representing all the application areas of the ITS domain are listed in Table 4.1 in terms of the organisational concept of intelligent agent development. ITS user services are derived from the view of various users (e.g. the travelling public, different system operators) to show what kinds of services and applications should be developed in ITS. Each user-service bundle can be considered as a suborganisation, and each user service within it represents more detailed suborganisations within a suborganisation. There are 33 user services in Table 4.1, and each service is grouped into one of eight bundles as possible service areas/composites: Travel and Traffic Management, Public Transportation Management, Electronic Payment, Commercial Vehicle Operations, Emergency Management, Advanced Vehicle Safety Systems, Information Management, and Maintenance and Construction Operations.

Table 4.1 - ITS User Services (Architecture Development Team, 2005)

User-Service Bundle	User Service
1. Travel And Traffic Management	1.1 Pre-trip Travel Information
	1.2 En-route Driver Information
	1.3 Route Guidance
	1.4 Ride Matching And Reservation
	1.5 Traveller Services Information
	1.6 Traffic Control
	1.7 Incident Management
	1.8 Travel Demand Management
	1.9 Emissions Testing And Mitigation
	1.10 Highway Rail Intersection
2. Public Transportation Management	2.1 Public Transportation Management
	2.2 En-route Transit Information
	2.3 Personalised Public Transit
	2.4 Public Travel Security
3. Electronic Payment	3.1 Electronic Payment Services
4. Commercial Vehicle Operations	4.1 Commercial Vehicle Electronic Clearance
	4.2 Automated Roadside Safety Inspection
	4.3 On-board Safety And Security Monitoring
	4.4 Commercial Vehicle Administrative Processes
	4.5 Hazardous Materials Security And Incident Response
	4.6 Freight Mobility
5. Emergency Management	5.1 Emergency Notification And Personal Security
	5.2 Emergency Vehicle Management
	5.3 Disaster Response And Evacuation
6. Advanced Vehicle Safety Systems	6.1 Longitudinal Collision Avoidance
	6.2 Lateral Collision Avoidance
	6.3 Intersection Collision Avoidance
	6.4 Vision Enhancement For Crash Avoidance
	6.5 Safety Readiness
	6.6 Pre-crash Restraint Deployment
	6.7 Automated Vehicle Operation
7. Information Management	7.1 Archived Data
8. Maintenance And Construction Management	8.1 Maintenance And Construction Operations

A range of technologies for information and communications can be considered to implement these services. However, since this research aims to support situation awareness and the collaboration of intelligent vehicles (and infrastructure) by DSRC, the target application area should be related to local vehicular communications. Emergency vehicles, such as police cars, fire engines, and ambulances, can be proactive participants of local emergency situations, which vary, ranging from traffic incidents to disasters to evacuations. Meanwhile, vehicles in the vicinity of an emergency vehicle have to be reactive in order to cooperate to resolve the situation or to avoid being another obstacle in the situation.

Therefore, exploring services related to proactive emergency vehicles and reactive vehicles around the emergency situations can be a good starting point to find a target application area. Situation-aware (proactive and reactive) characteristics of intelligent vehicle agents in emergency traffic situations are well-expressed in the user services in the fifth and sixth user-service bundles in Table 4.1, which deal with emergency management and advanced vehicle safety systems, respectively. Services in these two bundles are examined in the following two subsections to find the target application area by checking which services support the situation awareness and vehicular communication.

4.2.2 User services in the Emergency Management bundle

In the Emergency Management bundle, there are three user services: Emergency Notification and Personal Security; Emergency Vehicle Management; and Disaster Response and Evacuation (Architecture Development Team, 2005). This section examines these three user services in a time sequence, and then their operational functions, communication contents and types are reviewed to find a target application area that is suitable for the purpose of this research.

The three user services in the emergency management can be categorised into two parts based on a timeline from the occurrence of an incident or a disaster to the arrival of the response service on site as shown in Figure 4.2. The Emergency Notification and Personal Security user service belongs to the first part of the time sequence, which focuses on reducing the time from the occurrence of an incident or a disaster to the notification of an appropriate response body. The Emergency Vehicle Management user service and Disaster Response and Evacuation user service belong to the second part, which minimises the time required to provide the response to the site after the notification (Architecture Development Team, 2005).

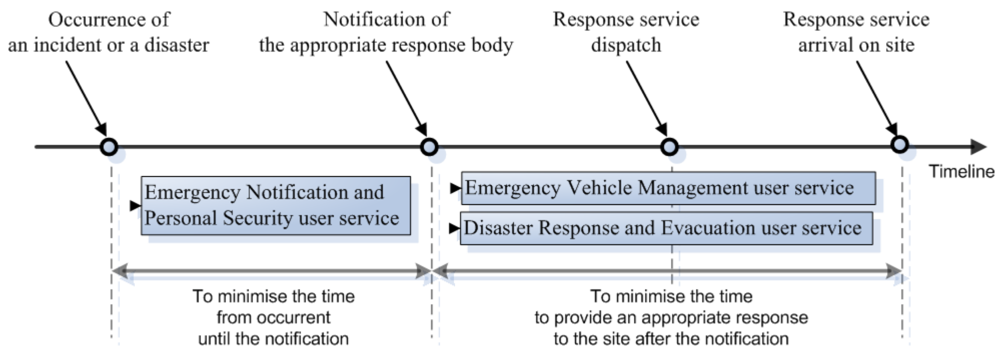

Occurrence of an incident or a disaster
Notification of the appropriate response body
Response service dispatch
Response service arrival on site
Timeline
Emergency Notification and Personal Security user service
Emergency Vehicle Management user service
Disaster Response and Evacuation user service
To minimise the time from occurrent until the notification
To minimise the time to provide an appropriate response to the site after the notification

Figure 4.2 - A timeline to provide an appropriate response to an emergency situation

Information for the first part of the time sequence focuses on giving notification of the occurrence of an incident or a disaster with its static location/area and type. Meanwhile, services in the second part of the time sequence support the response resources (personnel and/or vehicles) accessing the scene, so they require more dynamic information when describing a situation. In this context, dynamic information means that it contains and supports the dynamic movements of response vehicles and personnel. For instance, when an emergency vehicle is moving to the scene, its communicating targets and the communication contents (information) are also changing.

To find a user service requiring dynamic information in terms of communication subjects and contents in the Emergency Management bundle, each user service's subservices, operational functions, and communication types are described in Table 4.2. The first and third user services, which are the Emergency Notification and Personal Security user service and the Disaster Response and Evacuation user service, are based on the centre-related communications; the former gives notification of an incident or a disaster while the latter provides efficient information for safe evacuation from the scene. The second service, the Emergency Vehicle Management user service, requires various types of vehicle-related communications to assist emergency vehicles and their movements. It has three subservices (i.e. Emergency Vehicle Fleet Management, Route Guidance, and Signal Priority), and each subservice has different operational functions and communication types.

Table 4.2 - Subservices, operational functions, and communication types in the Emergency Management bundle (Architecture Development Team, 2005)

User Service	*Subservices* and operational functions	Communication type
Emergency Notification and Personal Security	*The Driver and Personal Security subservice and the Automated Collision Notification subservice*: • Notifying the occurrence of an incident with its location and type (e.g. mechanical breakdown, fire, non-injury accident, or injury accident)	Vehicle-to-Centre Traveller-to-Centre
	The Remote Security and Emergency Monitoring subservice, the Wide Area Alert subservice, and the Protect Sensitive Traveller Information subservice: • Broadcasting security-related information (e.g. threat alerts, severe weather warnings, natural and human-caused disasters, military operations, and civil emergencies) with its coverage (location/area)	Centre-to-Centre Centre-to-Infrastructure Centre-to-Vehicle Centre-to-Traveller
Emergency Vehicle Management	*The Fleet Management subservice*: • Providing communication connections between a dispatch centre and emergency vehicles • The dispatch centre makes real-time localisation of emergency vehicles possible • The centre can dispatch nearest emergency vehicles to an incident site so that response and arrival times of emergency service can be shortened resulting in improvement of dispatch efficiency	Vehicle-to-Centre Centre-to-Vehicle
	The Route Guidance subservice: • Assisting dispatcher and emergency vehicle driver in determining the optimal (in terms of time) route to reach the incident scene, and, if required, from the incident scene to a suitable hospital • Providing route guidance for directing the emergency vehicle driver to the destination	Centre-to-Vehicle in-vehicle
	The Signal Pre-emption subservice: • Providing the capability to pre-empt traffic signals on an emergency vehicle's route so that the emergency vehicle is nearly always presented with a green signal and makes it possible to minimise its travel time • Warning drivers of affected vehicles that an emergency vehicle is approaching	Vehicle-to-Infrastructure (or Centre-to-Infrastructure) Vehicle-to-Vehicle
Disaster Response and Evacuation	• Providing information for more efficient, safer evacuation for the general public in the vicinity of the disaster and the response personnel and resources accessing to the scene	Centre-to-Vehicle Centre-to-Traveller

In the Emergency Management bundle, communication technologies have primary roles to give notification of an incident and to share situational information with emergency vehicles as well as with affected vehicles and the road infrastructure. Although all user services in this service bundle are aimed at minimising the response time to an incident, the second user service (the

Emergency Vehicle Management user service) requires vehicle-related communications, the contents of which are directly related to vehicles' movements.

4.2.3 User services in the Advanced Vehicle Safety Systems bundle

In the service bundle of Advanced Vehicle Safety Systems, there are seven subservices: Longitudinal Collision Avoidance, Lateral Collision Avoidance, Intersection Collision Avoidance, Vision Enhancement for Crash Avoidance, Safety Readiness, Pre-crash Restraint Deployment, and Automated Vehicle Operation (Architecture Development Team, 2005). This section reviews these seven user services, which enhance drivers' perceptions of potential incidents with advanced electronics, communications, and processing and control systems.

User services of the Advanced Vehicle Safety Systems bundle are primarily based on sensors (e.g. video sensors, laser sensors, infrared sensors), information processing, and automatic vehicle control (vehicle steering, braking, and/or throttle actions) to minimise the number and severity of crashes and collisions (Architecture Development Team, 2005). However, as a subordinate information transmission, vehicle-to-vehicle and vehicle-to-infrastructure communications can support various situations, in which a vehicle is about to change lanes or cross an intersection, or a collision is impending. Table 4.3 represents subservices, functional operations, and possible subsidiary communication types in the Advanced Vehicle Safety Systems bundle. The first three user services allow the driver to perceive approaching vehicles to avoid longitudinal, lateral, and intersection collisions using in-vehicle sensors and/or vehicle-to-vehicle communications. The fourth user service (i.e. the Vision Enhancement for Crash Avoidance service) would be implemented by in-vehicle sensors during periods of poor visibility, such as night time or foggy conditions. Furthermore, the infrastructure to vehicle communication can assist intelligent vehicles (or the drivers) in complying with traffic signals and signs. Likewise, vehicle-to-vehicle and vehicle-to-infrastructure communications can assist intelligent vehicles (or the drivers) by providing the information about road conditions, anticipating an imminent collision, and allowing vehicle check-in to a dedicated highway lane, which are for the fifth, sixth, and seventh user services, respectively.

Table 4.3 - Subservices, operational functions, and communication types in the Advanced Vehicle Safety Systems bundle (Architecture Development Team, 2005)

User Service	*Subservices* and operational functions	Possible communication type
Longitudinal Collision Avoidance	*Rear-End Crash Warning and Control, Adaptive Cruise Control (ACC), Head-On Crash Warning and Control, and Backing Crash Warning and Control*: • Augmenting the driver's ability to avoid or decrease the severity of a two-vehicle collision in which vehicles are moving in essentially parallel paths prior to the collision, or one in which the struck vehicle is stationary	Vehicle-to-Vehicle Infrastructure-to-Vehicle
Lateral Collision Avoidance	*Lane Change/Blind Spot Situation Display, Collision Warning and Control, Lane/Road Departure Warning and Control*: • Enhancing the driver's perception to avoid collisions that arise when a vehicle moves out of its lane and moves in a forward direction	Vehicle-to-Vehicle
Intersection Collision Avoidance	*Warning of imminent collisions with crossing traffic, Warning of stop control (stop sign or traffic signal) in the intersection ahead*: • Augmenting the driver's ability to avoid or decrease the severity of collisions that occur at intersections	Infrastructure-to-Vehicle Vehicle-to-Vehicle
Vision Enhancement for Crash Avoidance	• Augmenting visually-acquired information in situations where driving visibility is low (such as night time or foggy conditions) in order to avoid collisions with other vehicles, fences, pedestrians, or obstacles • Assisting the driver in complying with traffic signals and signs	Vehicle-to-Vehicle Infrastructure-to-Vehicle (mainly implemented by in-vehicle sensors such as passive far infrared sensor, active radar, laser radar, etc.)
Safety Readiness	*Impaired Driver Warning and Control Override, Vehicle Condition Warning, and In-Vehicle Infrastructure Condition Warning*: • Providing drivers with a warning regarding their own driving performance, the condition of the vehicle, and the condition of the roadway as sensed from the vehicle	Infrastructure-to-Vehicle Vehicle-to-Vehicle
Pre-Crash Restraint Deployment	• Anticipating an imminent collision and activating passenger safety systems prior to the actual impact to reduce injuries and fatalities resulting from vehicle accidents	Vehicle-to-Vehicle (mainly implemented by sensors capable of detecting the rapid closing of distance between the vehicle itself and other vehicles before an actual impact occurs)
Automated Vehicle Operations (AVO)	• Improving the safety and efficiency of highway travel by moving suitably equipped vehicles under fully automated control (i.e. hands-off and feet-off operation) as part of the traffic flow by vehicle check-in to an AVO designated lane	Vehicle-to-Infrastructure Vehicle-to-Vehicle

In the Advanced Vehicle Safety Systems bundle, in-vehicle sensors and processors have primary roles to reduce the number and the severity of crashes. Nevertheless, vehicular communications still have subsidiary roles in this bundle to transmit situational information locally for the (potentially) affected vehicles. Although all the user services in this service bundle deal with sudden collisions and obstacles on the roadways or at an intersection, the first, second, third, and fifth user services (i.e. the three Collision Avoidance user services, and the Safety Readiness user service) are directly related to the information about the vehicle's movements and the road conditions.

4.2.4 The target application area of this research

In Section 4.2.2 and Section 4.2.3, the Emergency Management service bundle and the Advanced Vehicle Safety Systems service bundle were examined to find the user services that support the situation awareness and vehicular communication. A user service (i.e. the Emergency Vehicle Management user service) from the Emergency Management service bundle and four user services (i.e. the three Collision Avoidance user services and the Safety Readiness user service) from the Advanced Vehicle Safety Systems service bundle were chosen to represent user services requiring dynamic situational information (e.g. vehicles' movements) and vehicular communications (vehicle-to-vehicle and vehicle-to-infrastructure communications). This section specifies the target application area from the chosen user services in these two service bundles.

First, the Emergency Vehicle Management user service of the Emergency Management service bundle is examined. It has three subservices: the Fleet Management subservice, the Route Guidance subservice, and the Signal Pre-emption subservice. The Fleet Management subservice and the Route Guidance subservice can be implemented based on Centre-to-Vehicle communication and/or In-Vehicle communication. For these two subservices, emergency vehicles have a fixed communication connection with a dispatch centre to update their locations and to be informed regularly so that the communication subjects are static as long as emergency vehicles are inside the dispatch centre's coverage. Meanwhile, the Signal Pre-emption subservice can be implemented with *ad hoc* local communications from an emergency vehicle because the communication targets are dynamically changing as the emergency vehicle travels to the incident site or to the hospital. Whilst the emergency vehicle is travelling along its route, it interacts with traffic signal controllers and vehicles around it to minimise travel time to the scene and avoid potentially dangerous conflicts at the same time. The Signal Pre-emption subservice can be realised using vehicle-to-vehicle communication and vehicle-to-infrastructure communication, both of which DSRC is designed for. Of course, the dispatch centre can also control traffic signals for the emergency vehicle since the dispatch centre monitors the emergency vehicle's movements. However, the emergency vehicle has to communicate with vehicles that it is approaching anyway so local and direct communications (vehicle-to-infrastructure and vehicle-to-vehicle communication) from the emergency vehicle can be more effective.

Second, three Collision Avoidance user services and the Safety Readiness user service from the Advanced Vehicle Safety Systems service bundle are examined. For the three Collision Avoidance user services, vehicle-to-vehicle communications can be used to enhance vehicle's situation awareness by sharing vehicles' location and movement. For the Safety Readiness, vehicle-to-vehicle and infrastructure-to-vehicle communications can be used to share information about the conditions of a vehicle and of a road segment, respectively. Even though these user services are mainly implemented by electronics (e.g. sensors, processing and control systems), vehicular communications can be used to enhance drivers/vehicle agents' situation awareness. In addition, the situations of these user services are compatible with the situations of the Signal Pre-emption subservice of the Emergency Management bundle.

From two service bundles, subservices relating to the urgent situations containing vehicles' dynamic movements are selected as the target application area (Figure 4.3). In emergency situations, vehicular communications can enhance intelligent vehicles' (or drivers') situation awareness to resolve the situations. The Signal Pre-emption subservice of the Emergency Management service bundle represents emergency vehicles' proactive characteristics based on vehicle-to-infrastructure and vehicle-to-vehicle communications. As shown in Figure 4.3, seven subservices exist as part of the Advanced Vehicle Safety Systems service bundle, which represents the reactive characteristics of intelligent vehicles in emergency situations. These subservices are compatible with vehicle-to-vehicle and vehicle-to-infrastructure communications. The possible communication contents of the chosen target application area may include the vehicle's heartbeat data (e.g. vehicle's position, speed, direction of travel, and size) and the signal phase and timing data of traffic signals.

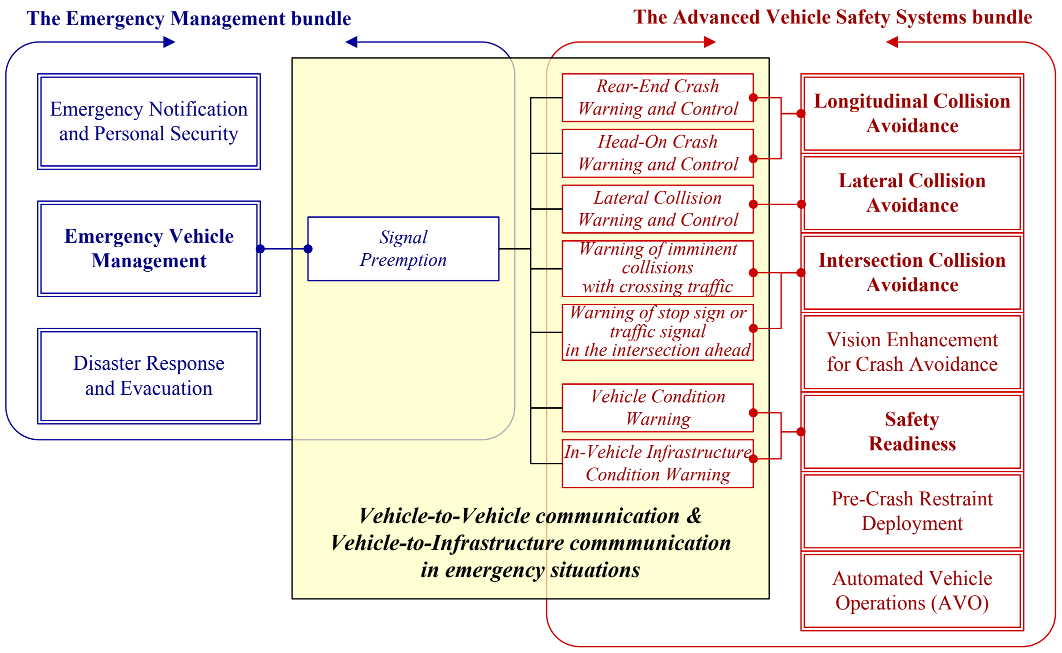

The Emergency Management bundle
The Advanced Vehicle Safety Systems bundle
Emergency Notification and Personal Security
Emergency Vehicle Management
Disaster Response and Evacuation
Signal Preemption
Rear-End Crash Warning and Control
Head-On Crash Warning and Control
Lateral Collision Warning and Control
Warning of imminent collisions with crossing traffic
Warning of stop sign or traffic signal in the intersection ahead
Vehicle Condition Warning
In-Vehicle Infrastructure Condition Warning
Vehicle-to-Vehicle communication & Vehicle-to-Infrastructure commmunication in emergency situations
Longitudinal Collision Avoidance
Lateral Collision Avoidance
Intersection Collision Avoidance
Vision Enhancement for Crash Avoidance
Safety Readiness
Pre-Crash Restraint Deployment
Automated Vehicle Operations (AVO)

Figure 4.3 - The target application area of this research

To cover all the ITS services, vehicle agents have to support several services simultaneously, and they need to cooperate, coordinate, and negotiate with each other to provide multiple services and achieve multiple goals. However, the aim of this research is to show the potential of real-time vehicular communications in safety-critical applications. Therefore, the target application area is chosen, focusing on vehicle agents' voluntary cooperation as individuals. The individual vehicle agents resolve an emergency situation by communications rather than their coordination and negotiation by a managing authority.

The next section will describe two DSRC scenarios representing situation-aware (proactive and reactive) vehicle agents and their local vehicular communications from the chosen target application area.

4.3 DSRC Scenarios

By 2004, more than 100 DSRC applications have been suggested, and the list is still expanding (Schnacke, 2004). In many cases, applications need several different interactions between vehicles and infrastructure in a similar setting. Even though specified applications cannot cover all possible situations and events that may occur, the process of describing situations for implementing an application is necessary. Scenarios can be the bridge between applications and situations since a scenario is ‘a rich and detailed portrait of a plausible future world’ presenting challenges and opportunities (The Futures Group, 1994, p.1). In this section, two scenarios are proposed to describe situations representing the target application area.

In system development, various types of scenarios (e.g. story, situation, simulation, sequence, etc.) are used for different purposes. In this research, scenarios are used to describe problematic traffic situations and to highlight the potential of DSRC applications that can resolve the situations. Therefore, for the discussion here, a scenario means a story-like dynamic situation that provides a narrated description of a sequence of events bringing traffic contexts in a futuristic situation with imagined snapshots. From the scenarios, a geospatial ontological framework is developed in Chapter 5, and then agent modelling and simulation is used to examine vehicular communications and the ontological framework in Chapter 6. Therefore, in this thesis, the term ‘simulation’ has a different meaning from the term ‘scenario’.

In a scenario, interactions among vehicles and road facilities are regarded as building blocks to describe a part of the situation from the perspective of vehicles and road facilities. Scenarios are very useful to find and categorise essential interaction components that can be used to build up a semantic framework for describing and resolving various situations.

For scenarios, the communication environment can be assumed to fulfil one of three conditions: first, every vehicle and road facility can communicate with every other; second, some vehicles and road facilities can communicate with each other; or third, only limited vehicles and road facilities can communicate with each other. Even though this research focuses on the first condition dealing with communicative vehicles, hybrid situations wherein communicative vehicles and non-communicative vehicles coexist have to be considered. This is because these situations are inevitable during the transition period towards an ITS environment. Even after the transition, there is a possibility that some communicative vehicles can experience device/system failure somehow, and in this case, the communicative vehicles have to be treated as non-communicative vehicles.

The following two sections describe two scenarios exploring possible communications and communication contents to resolve emergency situations. The first scenario outlines a situation of an approaching ambulance that represents the Signal Pre-emption subservice of the Emergency Management bundle. The second scenario describes a car breakdown situation relating to the Safety Readiness user service from the Advanced Vehicle Safety Systems bundle.

4.3.1 A scenario about an ambulance

The Signal Pre-emption subservice of the Emergency Management service bundle represents vehicle-to-infrastructure communications and vehicle-to-vehicle communications to provide the capability to pre-empt traffic signals and warn the drivers of affected vehicles around an emergency vehicle, respectively. The first scenario focuses on the vehicle-to-vehicle communications in an emergency situation where an ambulance broadcasts an alert message warning of its approaching to ask for priority over the vehicles around it. This scenario is based on the assumption that all vehicles have an OBE and can communicate with each other. The communication message must include the contents of the ambulance's movement, such as its location, speed, flight path vector (FPV), communication range, etc. (Table 4.4).

Table 4.4 - An example of emergency vehicle alert message (Michaels et al., 2010)

Item	Value
Lat	51.4115136
Long	-0.3127456
Speed	22.3 m/s
Heading in FPV	due east
Key Phrase	"emergency vehicles on roadway"
Heading Applied	east and westbound traffic
Extent	for 50 meters
Mass	2500 kg
Response Equip	Ambulance
Responder Type	ambulance units
Response Type	'01' = emergency
Response Details	Complex byte representing
SirenInUse	'10' = inUse
LightbarInUse	'10' = inUse
MultiVehicleResponse	'01' = singleVehicle

From the example shown in Table 4.4, an ambulance scenario was generated (Figure 4.4). In the scenario, an ambulance (`A1`) and another vehicle (`V1`) are following separate routes, which meet at an intersection (`i1`). If `A1` is in a hurry to get to the hospital and there are cars travelling towards the same intersection as `A1`, subsequently sharing the same road segments, `A1` can request priority over the other vehicles. Their interactions and movements are described with time stamps in Figure 4.5. The ambulance `A1` requests priority over cars towards `r3`. There could be vehicles in front of `A1` on `r1`, `r2`, and `r3`. In this case, there is one vehicle (`V1`) on `r2`, and `V1` gives priority to the ambulance (`A1`). Even though the scenario uses only an ambulance and a vehicle to show possible interaction between an emergency vehicle and the vehicles around it, the scenario can be extended with other vehicles and traffic signals for its implementation.

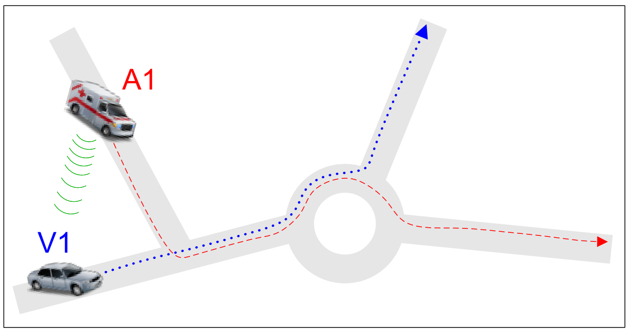
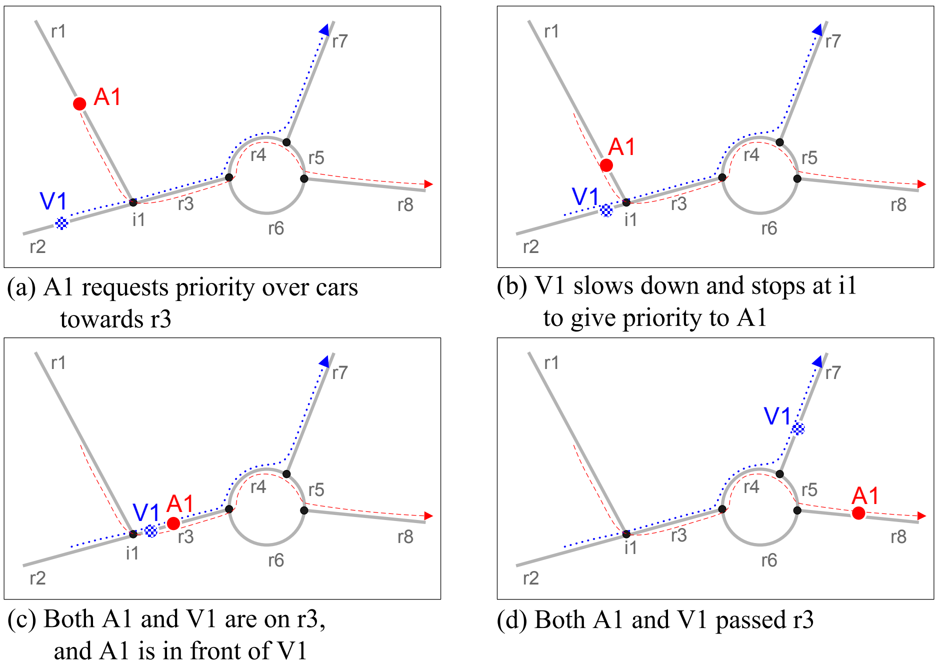


Figure 4.4 - A scenario of an ambulance communicating with the vehicles around it



(a) A1 requests priority over cars towards r3

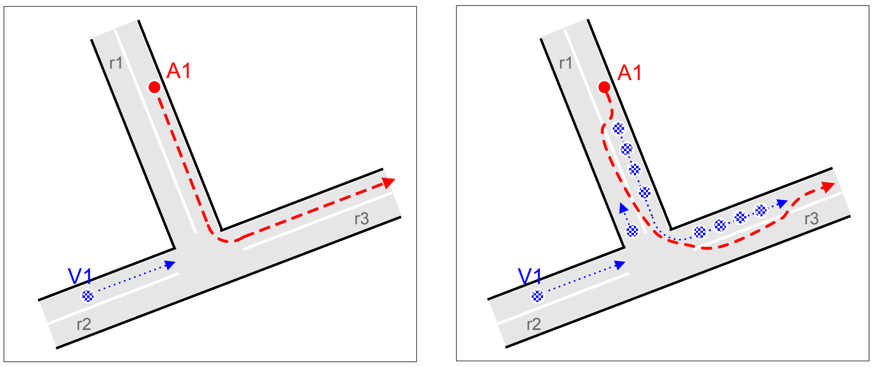
(b) V1 slows down and stops at i1 to give priority to A1

(c) Both A1 and V1 are on r3, and A1 is in front of V1

(d) Both A1 and V1 passed r3

Figure 4.5 - Snapshots of the ambulance scenario

When the roads are not crowded, the ambulance `A1` needs to consider only vehicles on the roads it is going to. However, if there is a traffic jam on the lane that `A1` is in, and the opposite lane is not too crowded, `A1` may use the opposite lane to move as quickly as possible in an emergency situation (Figure 4.6). In this case, `A1` has to be aware of vehicles in both directions and communication targets can be increased.



(a) Roads are not crowded

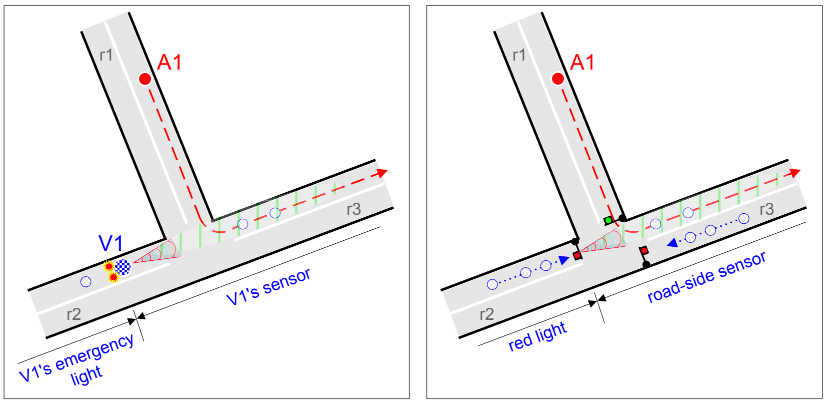
(b) One way traffic is blocked,
but the opposite way is fine

Figure 4.6 - Ambulance's expected route depending on the road status

The ambulance scenario is developed based on the assumption that all vehicles have at least a communication device and a position sensor, such as GPS, in order to share traffic situations based on their locations. However, as mentioned at Section 1.2, an ITS setting is composed of intelligent vehicles and roadside facilities, which have context-awareness, *ad hoc* networks, and smart sensors (Strang and Linnhoff-Popien, 2004). If the vehicular communication network cannot cover all vehicles and infrastructure, which is the second condition or third condition mentioned of the communication environment described in Section 4.3, other sensors, such as video camera, LIDAR, or radar sensor, can be used to identify vehicles in the vicinity and to supplement the limitation of partial communication.

Let us suppose a situation of the second condition where only some vehicles can communicate with each other at an intersection. In this case, even though `A1` and `V1` can communicate with each other, `A1` could not know whether there are other vehicles on `r2` and `r3` near the intersection. However, if `V1` has an optical sensor and processing power as well as a communication device, it can assist the ambulance partly with three actions (Figure 4.7 a). First, it can sense whether there are other vehicles ahead on `r2` and `r3`. Second, it can send information back to the vehicles behind by communicating and turning on the emergency lights so that it can control the cars behind it to give the ambulance priority. Thirdly, it can inform the ambulance that the vehicles behind it are in control and whether there are more cars in front of it on `r2` and `r3`. Therefore, the situation can be fully controlled when `V1` senses that there is no car ahead on `r2` and `r3`.

On the other hand, if ambulance `A1` is in the third condition of the communication environment, only limited vehicles and road facilities, such as ambulances and traffic controllers, can communicate with each other. In this case, if there are no traffic lights on the intersection, there is no way to assist the ambulance by communication. However, if there are traffic lights and a traffic controller on the intersection, the ambulance can request a green signal from the traffic controller. Then the traffic controller may present a green signal for the ambulance while it presents a red signal to vehicles on `r2` and `r3` (Figure 4.7 b). If there is a roadside sensor monitoring road `r3`, this can result in a case similar to the above situation in the second condition.



(a) An example of the second condition (b) An example of the third condition

Figure 4.7 - Sensing technologies assisting in traffic situations

Despite the fact that additional sensors can be used to describe a situation, the research adheres to the basic scenario, in which vehicles have only a communication device and a position sensor. Vehicles that have these two devices fulfil the minimum conditions to be considered as intelligent vehicles, and in this way, the research can focus on developing a reliable spatiotemporal data framework for vehicular communications.

4.3.2 Another scenario about a car breakdown on the motorway

As mentioned above, in some cases, communication technology is not enough to resolve a situation. In many cases, sensing technologies are essential for gathering information from the physical environment. Various sensors can be implanted into vehicles and road facilities in order to assist safe traffic flows on the road by capturing situational information from the real world environment, as described in Section 4.2.3. After vehicle and infrastructure agents receive the information, they can then share the information and cooperate with each other via communications to prepare for the situation and avoid being part of the problem. For example, if there is an icy road or an icy bridge ahead and a vehicle gets the information from its sensor directly or by communication with other vehicles that have already sensed the icy spot, they may change lanes to avoid the ice or slow down to minimise the risks in advance. If there is a car accident on the motorway, the accident scene can be a traffic obstacle. An icy road, a foggy area, and an accident spot are similar in the way that they can be obstacles to traffic flows on the road. The second scenario is developed to represent vehicle/infrastructure condition warning subservices, which are part of the Safety Readiness user service in the Advanced Vehicle Safety Systems service bundle.

In the second scenario, a car breakdown situation is depicted as a traffic obstacle on the motorway (Figure 4.8). There is a vehicle `B1` on the right lane of a two-lane motorway[17] for each direction, and a sudden problem occurs in vehicle `B1`'s engine (Figure 4.9a). So, the driver of the vehicle `B1` turns the emergency lights on and manages to pull the vehicle over into the left-hand lane of the motorway since there is no hard shoulder or breakdown lane on this road section of the motorway (Figure 4.9b). Then, the driver of the vehicle notifies a breakdown service provider of its condition and awaits the response. If the broken-down vehicle `B1` can broadcast its presence to the vehicles behind, the recipient vehicles (i.e. `V2`, `V3`, and `V4` of Figure 4.9c) can slow down or change lanes. Since the vehicle stops on the second lane, which is designed for large and slow-moving vehicles, such as goods vehicles and buses, it is hard for coming vehicles to realise the situation if the vehicles (e.g. `V4` of Figure 4.8) have an obstructed view because of large vehicles in front.

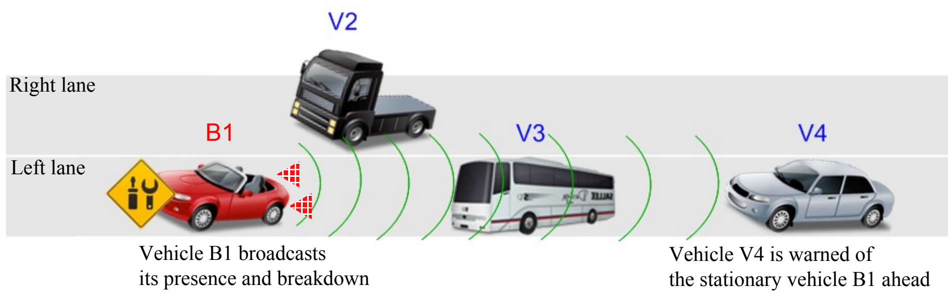
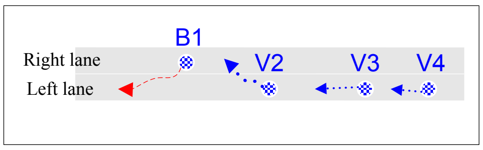
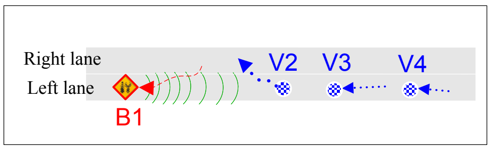


Figure 4.8 - A scenario of a broken-down car broadcasting its sudden stop

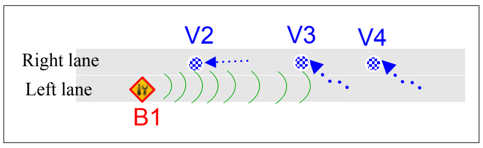
[17] Normally, there are more than two lanes (e.g. dual three-lane motorway, dual four-lane motorway), but a dual two-lane motorway is considered for a simple description of the vehicle breakdown scenario.



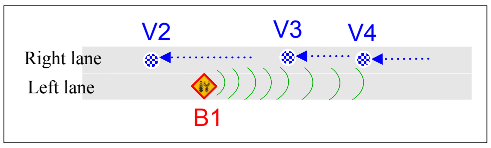
(a) A sudden problem occurs in B1's engine, and B1 pulls over on the left lane



(b) The broken vehicle B1 broadcasts its presence to the vehicles behind



(c) The recipient vehicles (V2, V3, V4) slow down and change lanes



(d) The recipient vehicles pass B1without sudden breaking or sudden lane changes

Figure 4.9 - Snapshots of the vehicle breakdown scenario

If a recipient vehicle also can communicate with others, it cannot only relay the broadcast message of the broken-down car, but can also communicate its sudden braking or sudden lane change to the vehicles heading towards it. Using vehicle-to-vehicle communications, vehicles can receive the situational information about an obstacle (i.e. the car breakdown in this scenario) and have more response time to slow down and change lanes without any sudden reactions. Even though this scenario contains the vehicle-to-centre communication to notify a breakdown service provider of the car breakdown, it is designed to highlight the vehicle-to-vehicle communications to broadcast/give warning of the broken-down car's presence to vehicles approaching the scene.

4.4 Spatiotemporal relations and location representation for the scenarios

Two scenarios were described in Section 4.3 representing the use of vehicle-to-vehicle communications to resolve emergency situations. The first scenario was about an ambulance, which travels much faster than other cars, while the second scenario was about a car breakdown, which is a sudden obstacle and danger that blocks rapidly moving vehicles on the motorway. Using communications, the ambulance in the first scenario could request priority over cars and traffic signals, and in the second scenario, vehicles heading towards the car breakdown spot could reduce their speed gently and prepare to stop and/or change lanes. In both cases, vehicles are informed about a situation beforehand via vehicular communications so that they have more response time to prepare for and take action to deal with the situation.

For both scenarios, communications are essential for vehicles and the infrastructure to share traffic situations. For the communication contents, vehicles' dynamic locations and their spatial relations (i.e. distance relations and directional relations) are crucial elements because their relative locations are key information to decide whether they are in the affected area of the emergency. The following two subsections outline spatial relations and linear referencing to support effective traffic interactions and location representation, respectively. Section 4.4.1 outlines how conventional traffic interactions can be extended by vehicular communications based on spatiotemporal relations among vehicles and the road infrastructure. Section 4.4.2 proposes a relative representation of vehicles' locations and movements for simple and intuitive communication contents. From the perspectives of spatial relations and location representation, situational information of the scenarios can be extracted as components of the communication contents.

4.4.1 Spatiotemporal relations for communication-based traffic interactions

This section describes how vehicular communications can support traffic interactions by generating spatiotemporal relations among vehicles and the road infrastructure. It starts with describing conventional traffic interactions without communications. Then, it demonstrates how traffic interactions can be extended by vehicular communications. Finally, it shows how spatial relations can be generated by vehicular communications.

4.4.1.1 Conventional traffic interactions without vehicular communications

Regardless of communication technologies, (drivers of) vehicles and the road infrastructure already cooperate, coordinate, and negotiate with each other following traffic situations and instructions. When a vehicle reduces its speed or is turning right at an intersection, it interacts with the vehicles around it by turning on the brake-lights or the indicator. It also follows traffic signals and traffic signs along the road. However, in emergency situations, emergency vehicles are exempt from several traffic rules and regulations by law.

For example, in the emergency situation of the first scenario, the drivers of the emergency vehicles (police car, fire engine, ambulance, etc.) can treat a red traffic light as a give way sign[18] and exceed the statutory speed limit[19]. Emergency vehicles override general traffic instructions, and most road users should allow emergency vehicles' priority. The problem is that if drivers' perception and response time for an emergency vehicle is not fast enough, the rapidly moving emergency vehicle may be involved in exposure to an incident/collision at an intersection or blocking by vehicles ahead. Even if an ambulance approaching an intersection is asking for priority over other vehicles by using its siren and strobe warning light, the traffic signal controller of the intersection is not aware of the ambulance's presence. The traffic lights at the intersection will still function in accordance with the already fixed cycle based on the hourly statistics of the traffic volumes of the road. Therefore, potentially affected vehicles heading to the intersection may experience a conflict between a green signal and the ambulance's approach.

Likewise, in the emergency situation of the second scenario, the broken-down car on the motorway can have a negative influence on the traffic flow and so cause a traffic jam. Following vehicles could slow down suddenly and become stuck in the traffic jam without knowing why. If the following vehicles have an obstructed view because of larger goods vehicles and coaches, they can misunderstand the situation and not try to change lanes until they get a clear view of the scene, which can exacerbate the situation.

4.4.1.2 Extended spatiotemporal relations via vehicular communications

As described in the scenarios, if vehicles (and road facilities) can send and receive situational information via communication technologies, they can request and respond to such information to improve the safety and efficiency of the traffic flow in real time and resolve various local situations.

To support traffic interactions based on vehicular communications, information about vehicles' movements, traffic signal's location, and their spatial relations at a specific time, has to be a part of the communication contents. There can be two kinds of spatial relations: static spatial relations and dynamic spatial relations. If there is a traffic light controller at an intersection and

[18] Regulation 36 of the Traffic Signs Regulations and General Directions 2002, http://www.legislation.gov.uk/uksi/2002/3113/regulation/36/made
[19] Section 19 of the Road Safety Act 2006, http://www.legislation.gov.uk/ukpga/2006/49/section/19

four road segments are connected to the intersection, the traffic light controller and the four road segments have spatial relations, which are not changing and are temporally static. Meanwhile, if there are vehicles on the road segments, spatial relations between the vehicles and the traffic light controller can be dynamic as vehicles are moving on the roads and their locations are spatially changing continuously. This kind of spatial relation is only valid in a specific time period when the vehicles are coming to the intersection that the traffic controller controls, so their dynamic spatial relations can be regarded as spatiotemporal relations.

Figure 4.10 describes binary relations between individuals (instances) of Vehicle, Route, RoadElement, Intersection, and TrafficLightController classes on a road network. For example, a vehicle has an origin, a destination, a current location, and its own route for a journey, while a traffic light controller is located near an intersection and controls the traffic lights for vehicles on the road towards the intersection. Basically, a binary relation between two objects means that an object has another object as a property so that it is also called as an object property. Since this research has focused on emergency situations on the road, most binary relations in Figure 4.10 represent spatial relations or spatiotemporal relations among vehicles and road facilities referring to the road geography and network.

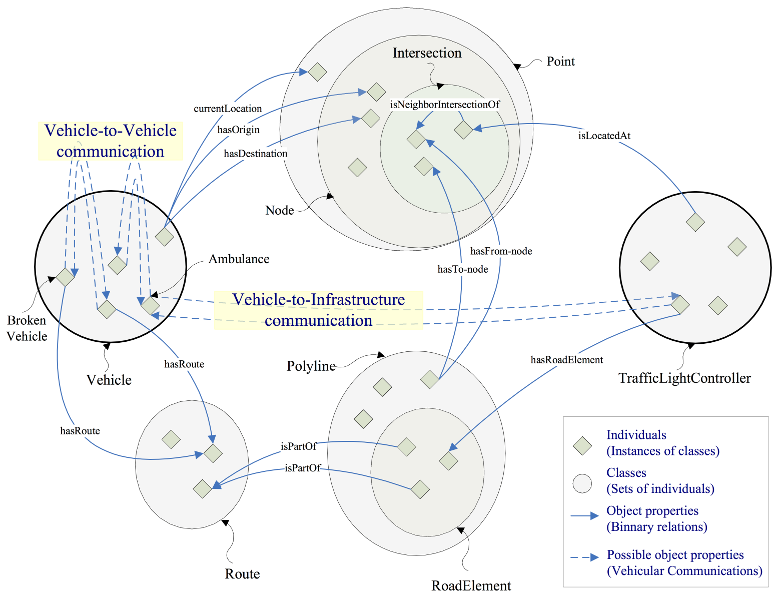


Figure 4.10[20] - An example of binary relations based on vehicular communications

[20] It follows the representation (i.e. legend) of OWL ontologies (i.e. individuals, classes, properties) that was written by Horridge et al. (2011).

There are various possible binary relations (dotted lines in Figure 4.10) between a vehicle and a traffic signal controller and between a vehicle and another vehicle. If the ambulance of the first scenario is heading towards an intersection at which a traffic signal controller controls the traffic lights, the vehicle-to-infrastructure communication link between the ambulance and the traffic light controller can be made. The ambulance needs to transmit a warning message to the traffic controller to request its signal pre-emption. Likewise, vehicle-to-vehicle communication links are necessary for the ambulance of the first scenario and the broken-down vehicle of the second scenario to send a warning message to potentially affected vehicles in the vicinity of the ambulance or the broken-down vehicle. It is obvious that the location and route information of the ambulance or the broken-down vehicle referring to the road geography and network is important for the recipient vehicles to find out whether to take action or not. If a vehicle receives a message from the ambulance, the vehicle needs to calculate its location relative to the ambulance. If the vehicle is in front of the ambulance and they have a shared route, the vehicle needs to take action when it is in the affected area. If a vehicle on the motorway receives an alert message from the broken-down vehicle, and it is located behind the broken-down vehicle, it also needs to take action.

To sum up, spatiotemporal relations among vehicles and the road infrastructure are essential to support vehicular communications. When vehicles share their locations and routes with reference to the road geography and network via vehicular communications, their spatiotemporal relations can be generated to represent the interrelation of the participant vehicles of the situations of the scenarios.

4.4.2 Relative location representation

In Section 4.4.1, the binary relations representing spatiotemporal relations among vehicles and road facilities are emphasised because an entire situation can be depicted by describing the spatially related participants of the situation. A binary relation between two vehicles can be generated when they communicate with each other to share information about their presence and compare their locations to decide whether the recipient vehicle is in the affected area of the situation and so take action or not. This section proposes a relative location representation to simplify the communication contents and split the geometric calculation from the decision-making process of vehicles.

To implement the first scenario, the ambulance needs to broadcast a communication message, and the recipient vehicles need to check whether they are in a situation to give way to the ambulance or not. For the second scenario, the car involved in the breakdown needs to broadcast a communication message about its presence. Of course, another vehicle or a road facility that senses the breakdown can broadcast a warning message on behalf of the broken-down vehicle. The recipient vehicles in both scenarios need to analyse the received message to check whether they need to slow down and give way to the ambulance or change lanes to avoid the broken-down vehicle. In both cases, the recipient vehicles need to analyse the message to check whether they are affected by the situation or not. In the emergency situations (i.e. an approaching ambulance in the first scenario and a car breakdown spot ahead in the second scenario), the recipient vehicles' interactions and responses depend on their spatiotemporal relations. Their location and route have to be compared with the location and route of the ambulance or broken-down vehicle to identify whether they can be affected by the emergency situation or not. This decision-making process of vehicles can involve geometric calculations, but the direct geometric distance (i.e. Euclidean distance) between two vehicles cannot represent their network distance (a.k.a. road distance) properly. Therefore, this research proposes a relative representation of geographical locations as an improved way to represent vehicles' spatiotemporal interrelation and support their interrelations.

In a transport network, in order to describe a situation, a location expression is required in emergency reporting, highway maintenance, etc. (Noronha and Church, 2002). Coordinates, street addresses, or landmarks can be used to express the location of a traffic situation. However, communicating parties in the transport domain have used linear referencing, in which a location is expressed relatively as a distance from a known reference point along a road centerline. This method is useful to dispatch a worker to a linear referenced location (e.g. a tunnel 3 km along a road from a reference point, rather than to a pair of coordinates, a street address, or a landmark) (Noronha and Church, 2002). In an ITS setting, vehicles and the roadside infrastructure are interconnected and form a vehicular network so that they can share each other's location-related messages in real time. Therefore, their relative locations can be used as communication contents to implement various DSRC applications, such as intersection collision avoidance, lane change warning, forward collision warning, or emergency vehicle warning (Schagrin, 2008).

The location and speed of the ambulance and vehicles in the first scenario can be represented relatively, i.e., in reference to local road segments and junctions. To simplify the scenario and focus on vehicles' longitudinal interrelation, the road network was treated as a one-dimensional linear space, and lane-level positioning and lane-related properties were not considered. The ambulance `A1`'s location (absolute coordinates) and speed (50 mph), (see Figure 4.5 a) can thus be referred to as 69.46 metres or 3.1 seconds from the intersection `i1`; while the vehicle `V1`'s location and speed (30 mph) is 60.72 metres or 4.5 seconds to the intersection. In this example, the vehicle `V1` needs to decelerate and stop before entering the intersection in order to give way to the ambulance because 1.4 seconds is not enough time to avoid a collision. In this way, the relative representation of a vehicle's location in an emergency scene helps each vehicle to understand the situation from an individual perspective so that each vehicle's decision-making process can be simplified by comparing the ambulance's and the vehicle's relative distance/time to a junction, which are scalar values, thus avoiding geometric calculations. It can be implemented if the ambulance broadcasts a warning message containing the relative distance and speed to a junction, and recipient vehicles around it calculate their own relative position and compare their relative location to the ambulance's relative location in turn. Of course, geometric calculations are necessary for each vehicle to generate relative information, but in this way, the decision-making process and geometric calculations can be separated.

4.5 Summary and discussion

This chapter described a process to extract the target application area from the ITS user services (Section 4.2), two scenarios representing the chosen target application area (Section 4.3), and spatiotemporal relations and location representation to decompose scenarios and build communication contents (Section 4.4).

First, Section 4.2 showed how a target application area can be selected from all the ITS user services. Two service bundles, which are the Emergency Management service bundle and the Advanced Vehicle Safety System service bundle, are extracted to represent intelligent vehicles' proactive and reactive characteristics based on vehicular communications. Then, several subservices in the two service bundles are chosen as the target application area of this research. The subservices in the chosen target application area deal with the dynamically moving vehicles based on vehicle-to-vehicle communications so that the target application area corresponds with the purpose of this research. This section also showed the general process used to extract a specific area (subsystem) from the whole system so that the process of this section can be used to extract different target areas representing the different characteristics of intelligent vehicles and infrastructure.

Second, in Section 4.3, two scenarios were developed to represent the chosen target application area. The first scenario describes a situation related to an ambulance while the second scenario depicts a breakdown situation on the motorway. The both scenarios contain dynamically moving vehicles and their communications. For the first scenario, the affected area of the ambulance is spatially dynamic as the ambulance is rapidly travelling on the road towards the incident spot or the hospital. The ambulance can be regarded as an obstacle for other vehicles because it disturbs the traffic flow. Similarly, the vehicles in the traffic flow can be regarded as obstacles for the ambulance if the ambulance's journey is delayed because the road is crowded. Meanwhile, the broken-down vehicle in the second scenario is spatially stationary. Consequently, the affected area of the breakdown can be spatially static; if vehicles are moving at free flow speeds and have enough time to respond to the breakdown, the breakdown will not cause a traffic jam. In this case, it is obvious that the broken-down vehicle is the obstacle for the vehicles approaching the breakdown spot. The two scenarios depict two different traffic situations representing the chosen target application area, in which vehicular communications may have a positive influence on resolving the situation.

Third, Section 4.4 showed how traffic interactions can be extended via vehicular communications and how spatial information about vehicles' locations and movements can be used as communication contents that describe traffic situations. The traffic situations of the scenarios can be decomposed into objects/agents and their properties including binary relations, and each component can be a part of the communication contents. In particular, vehicles and road facilities refer to the road geography and network to represent their locations and movements, so relative location representation (a.k.a. linear referencing) is proposed. The relative location representation simplifies the communication contents and separate geometric calculations from the decision-making process of the recipient vehicles.

For the process of constructing scenarios throughout the chapter, the analysis phase of the Gaia methodology (Section 3.2.2) was referred. The target application area and two scenarios are selected from whole ITS user services based on the sub-organisation concept of the methodology. The communication-based interactions and relative location representation are demonstrated to support intelligent vehicle's interactions and decision-making process resolving the emergency situations. In the scenario, resolving an emergency situation can be seen as an organisational rule. Therefore, in a perspective of the Gaia methodology, it can be said that in this chapter two sub-organisations are selected, and organisational rules are captured.

The scenarios developed in this chapter will be used for the ontology modelling in Chapter 5 and the agent modelling and simulation in Chapter 6. An ontological message model for vehicular communication is designed to support vehicles' processing and reactions through ontology querying and reasoning in Chapter 5. Chapter 5 will explain how ontology class structure and querying/reasoning patterns are created in the message model and a communication message can trigger neighbour vehicles' reaction to resolve the emergency situation. Conceptually, the process of Chapter 5 can be seen as the architectural design phase of the Gaia methodology. The ontological message model is replicated in a vehicle agent class for the simulation in Chapter 6. Building a vehicle agent model for the simulation can be seen as the detailed design phase of the Gaia methodology, and then the simulation is done instead of an implementation.

5. Geospatial ontological framework

5.1 Introduction

This chapter outlines the second stage of the methodological framework (Figure 1.3), that is, developing a geospatial ontology model for intelligent vehicles and infrastructure to share geosemantic information in an ITS environment. The ontology model contains a domain ontology, a task ontology, and an application ontology. Even though the domain ontology covers general concepts of the ITS domain, the task ontology and application ontologies focus on specifying the communications among vehicles and road facilities. Since vehicles and infrastructure are the most important components of the research, the ontology model is named 'VEIN' in reference to the 'VEhicle' and 'INfrastructure' classes.

In the following sections, the development process of the domain ontology, the task ontology, and the application ontology of the VEIN ontology model are explained. Then, the application ontology of the VEIN model is examined to demonstrate how vehicles and their interactions in the scenarios can be modelled. First, Section 5.2 outlines the domain ontology of the VEIN ontology model to describe situations in road transport based on the four main classes (vehicle, road infrastructure, centre, and traveller) of the ITS architecture and with reference to existing ontologies. Second, Section 5.3 deals with the task ontology to describe vehicle-to-vehicle communications and vehicle-to-infrastructure communications based on On-Board Equipment (OBE) and Road-Side Equipment (RSE), which are DSRC devices for vehicles and road facilities, respectively. For the domain ontology and the task ontology, this research provides only classes and their hierarchy in order to focus on the application ontology. Third, the application ontology of the VEIN model is developed in Section 5.4 to describe two scenarios with reference to the domain ontology and the task ontology. To satisfy the scenarios, some classes in the application ontologies are simplified and removed whilst properties and relations are added to describe the scenarios in more detail. Section 5.5 demonstrates how vehicles and their interactions in the two scenarios can be instantiated with the application ontology of the VEIN model. Some functions and query templates are made in the application ontology to assist vehicles' action and response in emergency situations. Lastly, Section 5.6 summarises and concludes the chapter.

5.2 Domain ontology of the ITS domain

In an ITS environment, there can be many applications, for example, arterial management, freeway management, crash prevention and safety, road weather management, or emergency management (Maccubbin et al., 2008). Even though this research focuses on short-range vehicular communication, the domain ontology should contain the general concepts of the ITS domain to ensure the opportunity that it can be extended for other ITS applications. Therefore, the process of building the domain ontology starts with a review of the four upper classes conceptualising the ITS architecture described in Section 1.3. Then, on top of the four upper classes, subclasses are captured and categorised from scenarios and existing transport ontologies in order to build classes and their hierarchy to describe a general ITS environment.

5.2.1 Four upper classes of the domain ontology

The four upper classes, which are `Vehicle`, `Infrastructure`, `Centre`, and `Traveller`, are derived from the four main classes of the ITS architecture as described in Section 1.3. It seems that these four classes are concepts that are compatible with the conventional road transport domain since they describe concepts that already exist, but there is a big difference between the conventional road transport domain and the ITS domain. In the ITS domain, vehicles, infrastructure, traffic centres, and travellers have most of the communication power. They can send and receive communication messages to share situational information so that they can be aware of situations beforehand and have a longer response time to take action when there is an emergency situation.

The subclasses of the four upper classes can affect each other based on their spatiotemporal relations. The possible interactions and relationships among the four upper classes are described in Table 1.1 of Section 1.3. A subclass in the `VEhicle` class and another subclass of the `INfrastructure` class can have a binary relation based on their locations at a particular time, and two subclasses belonging to different upper classes can communicate with each other to assist each other or to resolve situations. For example, a signal controller may send a stop sign violation warning to a vehicle in order to avoid a collision when the vehicle ignores traffic signals and tries to cross an intersection rapidly. A public transport vehicle may also request its priority to the traffic signal controller, in which case the traffic signal controller reschedules the traffic signals to resolve the situation.

A domain ontology should provide concepts (i.e. classes, terms) and the specific meaning of the concepts to represent a specific domain as a concept or term can have many different meanings. For example, the word 'vehicle' can be used for any manufactured device that moves people and goods, such as bicycles, motorcycles, trains, ships, boats, and aeroplanes. In this thesis, the word 'vehicle' refers to mobile machines that travel on the road in an ITS environment. So, the four upper classes of the domain ontology for the ITS domain are specified with the definition as follows.

- '`Vehicle`' indicates a mechanical road vehicle that has an engine to carry people or goods from place to place. Commuter cars, police cars, ambulances, and black cabs can be instances of this class. A vehicle may have an On-Board Equipment (OBE) for DSRC as a communication agent. `VEhicle` is a category and a super-class.
- '`Infrastructure`' indicates the fixed installations that allow vehicles to operate by providing the road environment. Road segments, intersections, traffic signs, traffic signals, and bridges can be instances of this class, and they may have a Road-Side Equipment (RSE) for DSRC. `INfrastructure` is a category and a super-class.
- '`Centre`' indicates a traffic centre that can communicate with vehicles, infrastructure, and travellers' mobile devices to transmit traffic information. For example, a dispatch centre can dispatch emergency vehicles to an emergency scene via mobile communication. A traffic centre can have a wire communication connection with road facilities because traffic centres and road facilities are both static and fixed spatially. `Centre` class is a category and a super-class.
- '`Traveller`' indicates (potential) road users such as pedestrians, drivers, or passengers of a vehicle who may have a mobile device to communicate with vehicles, the road infrastructure, and traffic centres. For example, a traveller can notify a traffic centre of a car accident to help them to dispatch emergency vehicles and resolve the situation. `Traveller` is a category and a super-class.

Even though this research focuses on the two scenarios described in Section 4.3, four upper classes are developed to model general concepts in the ITS domain. Instances of the `Vehicle` class and the `Infrastructure` class are subjects of communications. They are also instances of the road vehicles and the physical environment respectively to provide the road services to the road users that are instances of the `Traveller` class. The `Centre` class can be seen as a subclass of the `Infrastructure` class because it is also part of the road infrastructure that is providing the road service. However, in this research, the `Centre` class is defined as an upper class because it is the major subject of central communications providing general information (e.g. car accidents and congestions) on the road using radio and digital signboards.

Based on the four upper classes (`Vehicle`, `Infrastructure`, `Centre`, and `Traveller`), subclasses are generated. Even though some environmental events may affect the road environment and the road service, environmental classes are not considered for this research into vehicular communication since the environmental states can be translated as road states (e.g. rainfall to wet road condition). Meanwhile, rail transport is another major transport system on land, and road transport and rail transport sometimes intersect. So, some classes are added in the domain ontology to represent road facilities for the spots on which two transport systems intersect (e.g. railway crossing).

5.2.2 Referring existing domain ontologies

Developing an ontology involves following several steps: determining the domain and scope of the ontology, defining classes and their class hierarchy, defining the properties of the classes and the relations among the classes, creating instances of the classes, and creating rules/axioms (Noy and McGuinness, 2001). When defining the classes and their hierarchy for the VEIN domain ontology, some existing ontologies and terms were reviewed and used to represent situations of the two scenarios.

The VEIN domain ontology refers to GeoOWL (Lieberman et al., 2007), which describes a feature's location or geometry with simple coordinates. In GeoOWL, `where` property is used as an object property, in which the domain is `Feature` class and the range is `Geometry` class. Therefore, the four upper classes of the domain ontology described in Section 5.2.1 (i.e. `Vehicle`, `Infrastructure`, `Centre`, and `Traveller`) are defined as subclasses of the `Feature` class, so that they can describe their locations or shapes by using geometry (Figure 5.1).

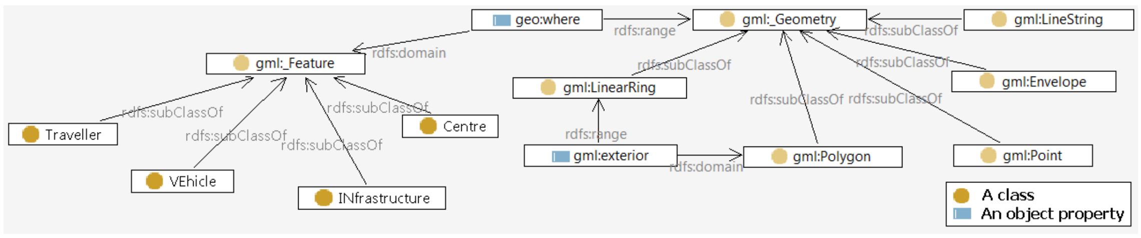


Figure 5.1 - The feature class and its subclasses in the VEIN domain ontology

Subclasses of the `VEhicle` class are captured from the ITS architecture of the U.S. Department of Transportation to describe their functions (Architecture Development Team, 2007). Some classes of OTN (Lorenz et al., 2005) are reused for subclasses of the `INfrastructure` class. OTN has many classes and complex relations, but OTN classes such as `Land Cover and Use` class and `Railways` class are not used, since the scenarios here describe on-road situations. In addition, some OTN classes and properties are simplified and modified in the VEIN domain ontology.

Some Ordnance Survey Ontologies containing object properties (i.e. `SpatialRelations` and `NetworkRelations`) are also referred to in order to describe the road connectivity and the relative location of a vehicle or a road facility on the road (Figure 5.2). Object properties can have various characteristics; for example, they can be symmetric, transitive, inverse, etc. For example, the `isConnectedTo` property of the `NetworkRelations` ontology is a symmetric property (`owl:Symmetric`), which means that if `X` is connected at `Y`, then `Y` is also connected at `X`; the `SpatialRelations:isLocatedBehind` property may be stated to be transitive (`owl:Transitive`), which means if `X` is located behind `Y` and `Y` is located behind `Z`, then `X` is located behind `Z`. A property `isLocatedInFrontOf` can be an inverse property (`owl:inverseOf`) of the `SpatialRelations:isLocatedBehind` property.

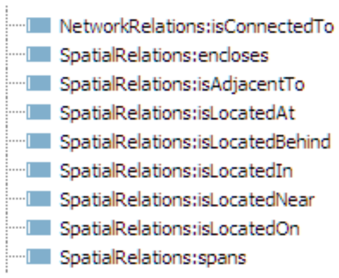

NetworkRelations:isConnectedTo
SpatialRelations:encloses
SpatialRelations:isAdjacentTo
SpatialRelations:isLocatedAt
SpatialRelations:isLocatedBehind
SpatialRelations:isLocatedIn
SpatialRelations:isLocatedNear
SpatialRelations:isLocatedOn
SpatialRelations:spans

Figure 5.2 - Ontology properties (from http://www.ordnancesurvey.co.uk/oswebsite/ontology/)

These properties may have restrictions on which values can be used (`owl:allValuesFrom`, `owl:someValuesFrom`) or how many values can be used (`owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality`). For instance, if on the road there are a traffic light controller and traffic lights, their relations can be described with a property (`controls`). In this case, the `controls` property can have restrictions; the domain has to be the traffic light controller class (`owl:allValuesFrom Traffic_Light_Controller`), the range has to be the traffic light class (`owl:allValuesFrom Traffic_Light`), and a traffic controller must control at least one traffic light (`owl:minCardinality` is 1).

5.2.3 Classes and their hierarchy in the domain ontology

The VEIN domain ontology is developed based on the four upper classes and existing ontologies/concepts to describe the four main components of the ITS architecture. Classes and their hierarchy in the domain ontology are shown in Figure 5.3. The `Traveller`, `Centre`, `Vehicle`, and `Infrastructure` classes are subclasses of the `Feature` class so that points, lines, and polygons can be used to represent the locations or shapes of the instances of these classes. The `Traveller` class is an equivalent class to the `RoadUser` class that contains various road users, such as passenger, pedestrian, cyclist, etc. The `Centre` class describes various traffic operation systems (i.e. administrations, managements, and services) including signalling operations, safety managements, and emergency managements. The `VEhicle` class indicates a mechanical road vehicle that has an engine and an OBE, and it is the super class for emergency vehicles, private vehicles, public transport vehicles, etc. The `INfrastructure` class refers to road facilities that have an RSE, and it is a category for the roads themselves, the road furniture, and the road structures. The domain ontology is for general descriptions of a road environment that is based on the four upper classes of the ITS architecture. The following two sections will describe the task ontology and the application ontology for the vehicular communications and the situations of the two scenarios, respectively.

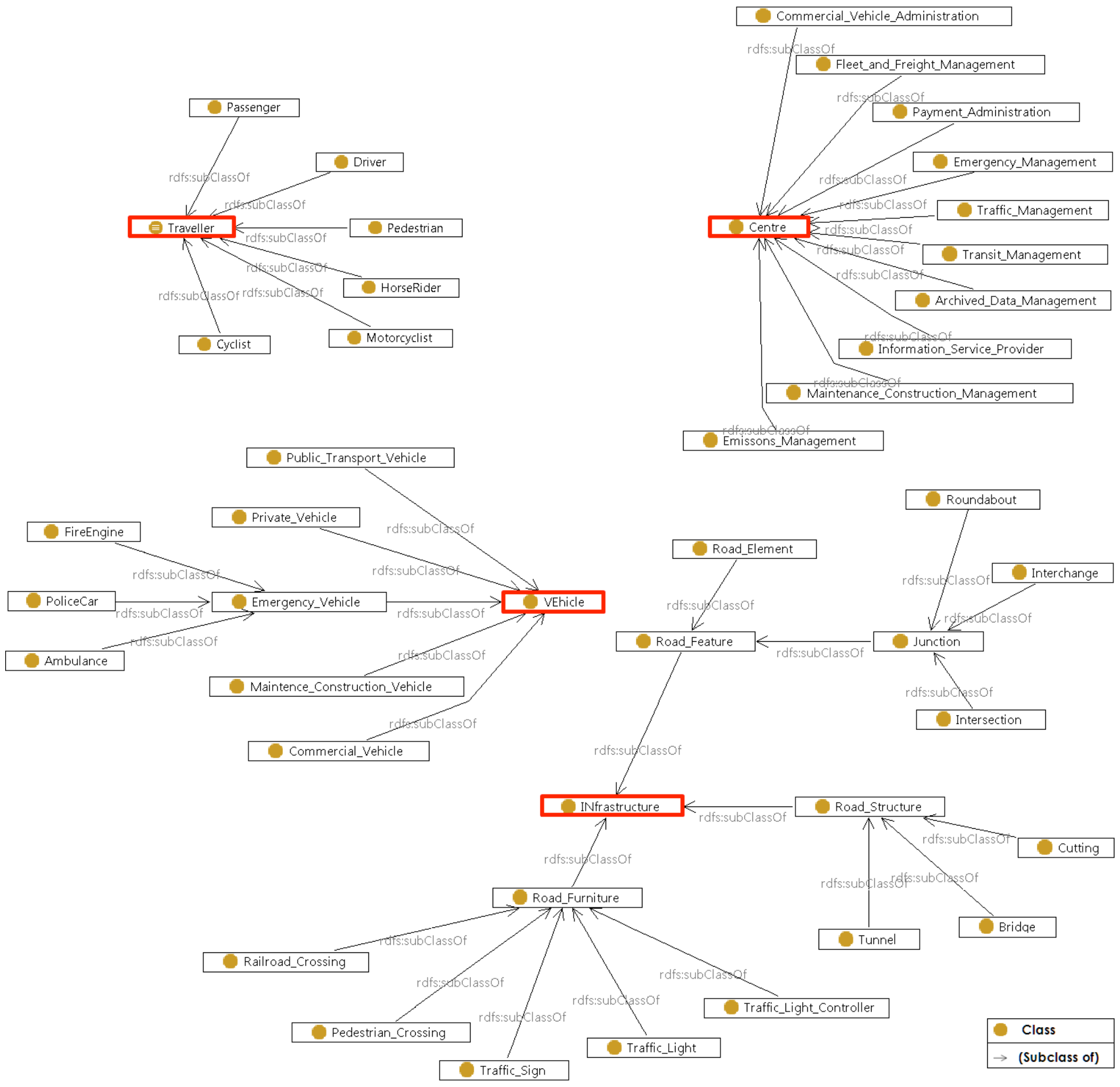


Figure 5.3 – Four classes of the VEIN domain ontology

5.3 Task ontology for DSRC

This section describes the concepts and classes relating to the communication task of vehicular communications, namely, DSRC. Two communication agent classes for vehicles and infrastructure are the main subjects of possible communications for the developed scenarios. As communication nodes, subclasses of the `Vehicle` class have an OBE as an object property, while the subclasses of the `INfrastructure` classes have an RSE. Their communication messages are also categorised into four subclasses. The main classes of the task ontology are extracted from DSRC communication flows among OBEs and RSEs, and then some general classes for communication events, communication processes, communication nodes and links are also generated as classes in the task ontology.

5.3.1 Communications between vehicles and road infrastructure

In a vehicular environment, there may be dynamic phenomena relating to moving vehicles, such as traffic signals, traffic jams, car accidents, etc. Some traffic events, such as traffic signals, are necessary to control traffic flows and provide efficient traffic services, while other traffic events, such as car accidents, have negative effects on traffic flow. Sections 4.4 already described specific situations of the scenarios to demonstrate how vehicular communications can extend traffic interactions and have positive effects on traffic flow. Since vehicles and traffic lights have different locations and signals at a time respectively, they can share their dynamic properties via communications to improve safety in urgent situations.

Figure 5.4 shows DSRC communications in traffic situations, in which emergency vehicles are involved, extracted from the architecture flows of the U.S. ITS architecture (Architecture Development Team, 2012). Of course, communications for parking management or toll collection are possible, but they are omitted in the figure in order to focus on the communications for vehicle safety. In the figure, a dotted line means an interaction between vehicles while a line means an interaction between a vehicle and a road facility. To share traffic situations, communication messages can contain emergency vehicle alerts, signal pre-emption requests, intersection status, etc. (Michaels et al., 2010).

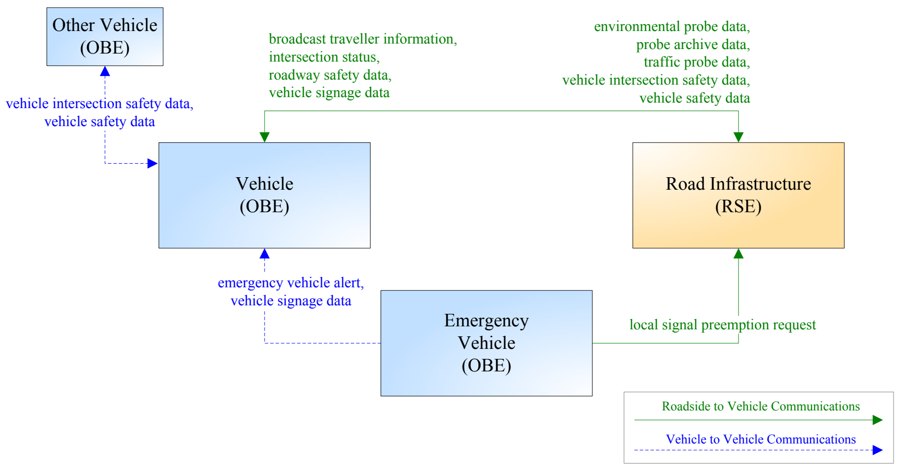

Other Vehicle (OBE)
vehicle intersection safety data, vehicle safety data
broadcast traveller information, intersection status, roadway safety data, vehicle signage data
environmental probe data, probe archive data, traffic probe data, vehicle intersection safety data, vehicle safety data
Vehicle (OBE)
Road Infrastructure (RSE)
emergency vehicle alert, vehicle signage data
Emergency Vehicle (OBE)
local signal preemption request
Roadside to Vehicle Communications
Vehicle to Vehicle Communications

Figure 5.4 - Architecture flows for vehicle-to-vehicle communication and vehicle-to-infrastructure communication

From the vehicle-to-vehicle communications and vehicle-to-infrastructure communications described in Figure 5.4, some ontology classes for communication agents and communication messages are extracted. The `Vehicle(OBE)` class and the `Infrastructure(RSE)` class are defined for vehicles that have an OBE and road infrastructure that has an RSE, respectively. Additionally, communication messages among vehicles and infrastructure are categorised into four message types: alert message (i.e. warning message), probe message, request message, and safety message (Figure 5.5).

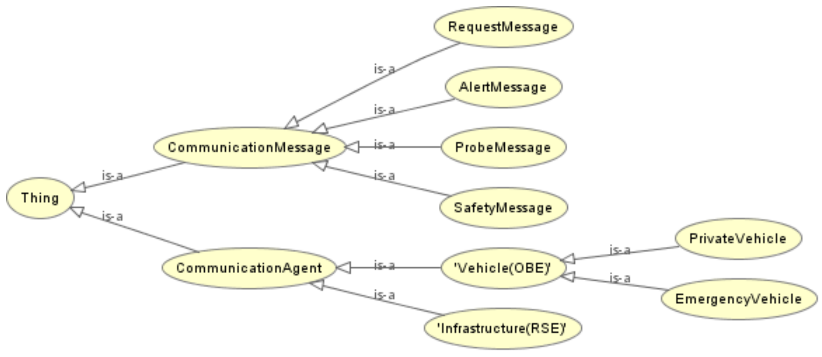


Figure 5.5 - Classes extracted for the task ontology

5.3.2 Five upper classes of the task ontology

Based on the communication agents and communication messages developed in the previous section, the task ontology is extended to support communications for four communication agents, namely, `Vehicle`, `Infrastructure`, `Centre`, and `MobileDevice`. The first three classes are the same as the classes in the domain ontology because each of them is already integrated with a communication equipment. Otherwise, the `Traveller` class of the domain ontology cannot be integrated with the `MobileDevice` class because the `Traveller` class is a class for human agents, and it is impossible to integrate them physically. Instead, a traveller can have a handheld device for communications. Therefore, in the task ontology, the `MobileDevice` class is created to represent travellers' handheld devices.

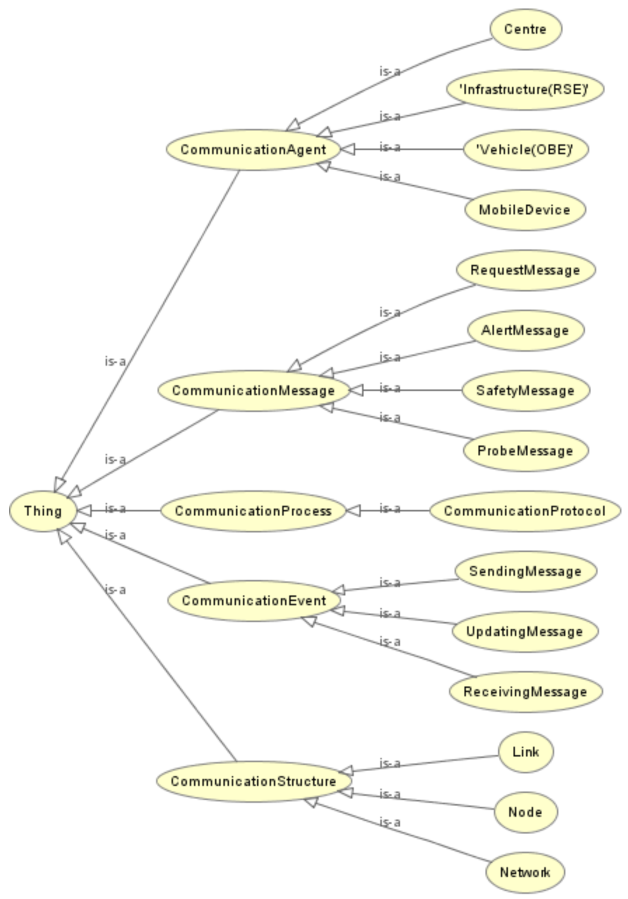


Figure 5.6 - Five upper classes and their subclasses in the task ontology

To describe communications among communication agents, there are four more classes: `CommunicationMessage`, `CommunicationEvent`, `CommunicationProcess`, and `CommunicationStructure` (Figure 5.6). Communication agents share situational information by using communication messages. Various contents can be included in the messages, such as a request, a warning, etc. Based on message types, several subclasses are defined in the `CommunicationMessage` class. When a communication agent, such as a vehicle or a road facility, generates information about a traffic situation, it has to share the information by sending and receiving messages, which are defined in the `CommunicationEvent` class. If the communication is a type of broadcasting, it is not necessary to build a communication network, but the case can arise where two communication bodies need to maintain the communication link. For example, the communication links between a dispatch centre and emergency vehicles are necessary. Communication node, communication link, and communication network are defined as subclasses of the `communication structure` class, while the `CommunicationProcess` class represents the low-level protocols of the communication. Even though the task ontology is designed to cover the general communication tasks in an ITS environment, the low-level protocols and processes of the communication are beyond the scope of this research because of the focus on the semantic contents of the communication among vehicles and road infrastructure.

In conclusion, five upper classes were proposed as the task ontology of the VEIN model to represent vehicular communications. In the next section, based on the domain ontology and the task ontology, the application ontology of the VEIN model is described to implement vehicular communications in the situations depicted in the two scenarios. While the domain ontology and the task ontology focus on the classes and their hierarchy, the application ontology deals with the properties and relations of classes to instantiate intelligent vehicles and their interactions.

5.4 Application ontology of the VEIN model

To describe the aforementioned two scenarios in the application ontology, geographic objects, such as road elements, junctions (e.g. intersections, roundabouts), and vehicles, are defined as classes of the ontology. The ontology also needs to have properties and relationships (hierarchical relations and semantic relations) between classes to provide the context of the scenario. In addition, some SPIN functions and templates are used to update instances of the application ontology representing traffic situations from the communication messages. This section explains classes, properties, relations, SPIN functions, and the SPIN template to instantiate the vehicles of the scenarios and to implement their interactions via vehicular communications.

5.4.1 Classes and their hierarchy of the application ontology

The `Vehicle` and `Infrastructure` classes are subclasses of the `Feature` class, so the instances of these classes have the `where` property to represent vehicles' locations and road facilities' geometries (locations and/or shapes). However, as this research follows the relative location representation to separate the geometric calculations from the vehicle's decision-making process, the application ontology is developed to conceptualise the two main classes (`VEhicle` and `INfrastructure`) and their properties and relations to describe the vehicles in the traffic situations of the scenarios and their communications (Figure 5.7). A partial route of a vehicle, an instance of the VEhicle class, can be represented with a list of upcoming junctions (comingJunctions), and the vehicle's location and direction can be described with the current road element (currentRoadElement), the next junction (nextJunction), and the distance remaining to the next junction (remainingDistanceToNextJunction). The road connectivity can be also described with the startsAt, endsAt, and isConnectedTo properties. The controls property can be used to describe relations between a traffic light controller and the traffic lights while the serves property can describe relations between traffic lights and road elements. This application ontology defines the necessary subclasses, properties, and relations for the VEhicle and INfrastructure classes only for the two scenarios, but it may be extended to support other scenarios in the future.

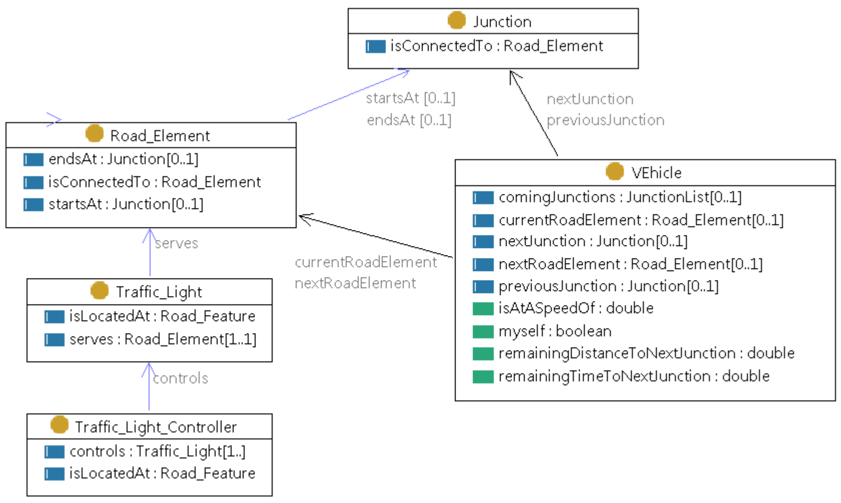

Junction
isConnectedTo : Road_Element
startsAt [0..1]
endsAt [0..1]
nextJunction
previousJunction
Road_Element
endsAt : Junction[0..1]
isConnectedTo : Road_Element
startsAt : Junction[0..1]
VEhicle
comingJunctions : JunctionList[0..1]
currentRoadElement : Road_Element[0..1]
nextJunction : Junction[0..1]
nextRoadElement : Road_Element[0..1]
previousJunction : Junction[0..1]
isAtASpeedOf : double
myself : boolean
remainingDistanceToNextJunction : double
remainingTimeToNextJunction : double
serves
currentRoadElement
nextRoadElement
Traffic_Light
isLocatedAt : Road_Feature
serves : Road_Element[1..1]
controls
Traffic_Light_Controller
controls : Traffic_Light[1..]
isLocatedAt : Road_Feature

Figure 5.7 - UML-like class diagram of the VEIN application ontology

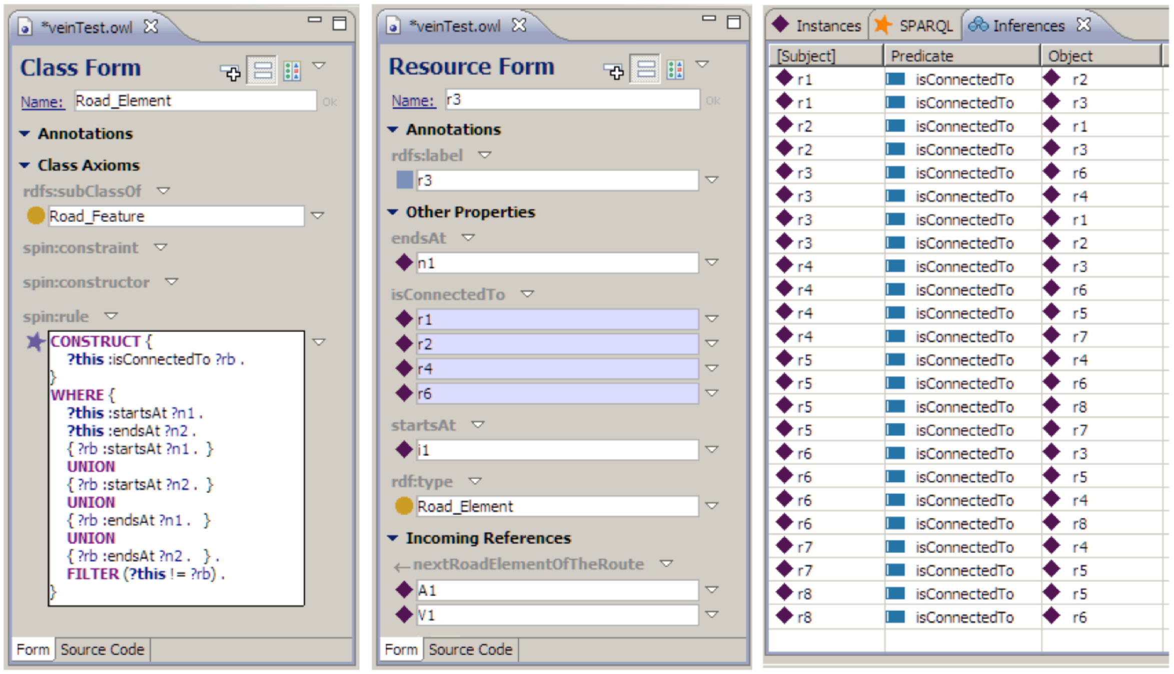
We have given a description of key classes, properties, and relations for the vehicle and infrastructure classes. The defined properties describe the relative locations of vehicles and road facilities, and they can also be used for vehicular communications to share information about traffic situations and interact with each other. Based on these properties and relations, the next section explains how to build a road network topology that is necessary for vehicles' routes.

5.4.2 Road element's connectivity

The road network is the basic road environment and the essential reference to describe vehicles' locations and route since vehicles are travelling on the road. It can be built from the road geometry of all navigable roads that can be extracted from road restriction information. This section simply shows how the road connectivity can be made from the road geography in general so that road restrictions are not considered.

To build the road network as a basic road environment for the scenarios, the link-node topology of the road network was inserted using `startsAt` and `endsAt` properties, in which the domain is `Road_Element` class and the range is `Junction` class. Since the range of the `startsAt` and `endsAt` properties is an object (i.e. an instance of `Junction` class) rather than a data type (i.e. int, double, String, etc.), these properties are defined as object properties (`owl:ObjectProperty`). These properties are also defined as functional properties (i.e. `owl:FunctionalProperty`), which can have only one unique value for each instance. So, every instance of the `Road_Element` class has only one starting junction and only one ending junction as the value of the `startsAt` and `endsAt` properties, respectively. In addition, there is a property (`isConnectedTo`) to represent the link-link topology of the road network. The triples using the `isConnectedTo` property can be generated automatically from the link-node topology (i.e. `startsAt` and `endsAt` properties). Figure 5.8 shows a SPIN rule in the `Road_Element` class to infer road elements' connectivity and its results. The SPIN rule is basically a SPARQL CONSTRUCT query, in which the system variable `?this`, UNION, and FILTER were used for the current instance of the class and the OR operation, and to avoid self-reference, respectively. This research assumes that the road network is static, so this SPIN rule was used just once, and inferred triples containing the `isConnectedTo` property were inserted as asserted triples into the ontology model.[21]

[21] Ontologies may contain asserted triples, which assert the properties of individuals, as ground facts. A SPIN rule is used to generate new triples (i.e. inferred triples) based on information of existing triples (i.e. asserted triples) by inferrencing. In the view of inferencing, asserted triples are the input of the inferencing process and inferred triples are the output of the inferencing process. The inferred triples represent the link-link topology of the road network, and it is part of ground facts in this research, so they are treated as asserted triples.



```
CONSTRUCT {
   ?this :isConnectedTo ?rb .
}
WHERE {
   ?this :startsAt ?n1 .
   ?this :endsAt ?n2 .
   { ?rb :startsAt ?n1 . }
   UNION
   { ?rb :startsAt ?n2 . }
   UNION
   { ?rb :endsAt ?n1 .  }
   UNION
   { ?rb :endsAt ?n2 .  } .
   FILTER (?this != ?rb) .
}
```

[Subject]	Predicate	Object
r1	isConnectedTo	r2
r1	isConnectedTo	r3
r2	isConnectedTo	r1
r2	isConnectedTo	r3
r3	isConnectedTo	r6
r3	isConnectedTo	r4
r3	isConnectedTo	r1
r3	isConnectedTo	r2
r4	isConnectedTo	r3
r4	isConnectedTo	r6
r4	isConnectedTo	r5
r4	isConnectedTo	r7
r5	isConnectedTo	r4
r5	isConnectedTo	r6
r5	isConnectedTo	r8
r5	isConnectedTo	r7
r6	isConnectedTo	r3
r6	isConnectedTo	r5
r6	isConnectedTo	r4
r6	isConnectedTo	r8
r7	isConnectedTo	r4
r7	isConnectedTo	r5
r8	isConnectedTo	r5
r8	isConnectedTo	r6

(a) an inference rule in the `Road_Element` class (b) the (grey-coloured) values are generated automatically (c) a set of new triples generated based on the rule

Figure 5.8 - A SPIN rule and new triples for road elements' connectivity

As we have seen so far, the SPIN rule using the `isConnectedTo` property demonstrates how indirect relations (i.e. link-link topology) of the road network can be made from the direct information (i.e. link-node topology) automatically. It also shows that SPIN rules can be used not only to extract information from the developed dataset, but also to construct further information for the dataset itself in the data construction phase. The next section outlines vehicles' properties and relations to represent their dynamic locations with reference to the road network.

5.4.3 Vehicles' location and route information

Vehicles are the main road users that travel fast on the road, and they are the essential components of dynamic road situations. From the perspective of geographic information, a vehicle's dynamic movements can be represented by its current location and route information. Vehicles' location and route information can be used to share their presences via vehicular communications in order to improve their traffic interactions and resolve traffic situations.

In order to store a vehicle's absolute coordinates, speed, route, and relative information, some object properties were designed, and these can be categorised into three parts (Table 5.1). The first part is the vehicle's absolute location and speed. The second part describes the vehicle's route information, referencing road elements and junctions. The property `comingJunctions` can have a junction list (`rdf:List`), which has five junctions to represent a vehicle's partial route. The third part represents the vehicle's relative location, which depends on the previous two parts.

Table 5.1 - Properties/relations related to each vehicle's movement

Purpose	Property	Domain	Range
Vehicle's absolute location and speed	geo:where	VEhicle	gml:_Geometry
	vein:isAtASpeedOf	VEhicle	xsd:double
Vehicle's route information	vein:previousJunction	VEhicle	Junction
	vein:comingJunctions	VEhicle	JunctionList
	vein:nextJunction	VEhicle	Junction
	vein:currentRoadElement	VEhicle	Road_Element
	vein:nextRoadElement	VEhicle	Road_Element
Vehicle's relative location	vein:remainingDistanceToNextJunction	VEhicle	xsd:double
	vein:remainingTimeToNextJunction	VEhicle	xsd:double

Table 5.2 shows how a vehicle describes its absolute location and route information in the VEIN application ontology. The `geo:where` property is used as an object property to describe its coordinates, and several object properties (e.g. `previousJunction`, `nextJunction`, `currentRoadElement`, and `nextRoadElement`) are used to describe its route information. In Table 5.2, the vehicle `V1` has object values `vein#n97` and `vein#i1` for the `previousJunction` and `nextJunction` properties, respectively. It means that there is a URI reference named 'vein', and it has fragment resources such as `n97` and `i1` representing a node and an intersection on a road network. The vehicle `V1`'s two properties mean that the vehicle already passed the node `n97` and moves towards the intersection `i1`.

Table 5.2 - A vehicle's properties and related instances

Description	Source code

Vehicle V1's current coordinates and route information	`<vein:Private_Vehicle rdf:ID="V1">`
	`  <geo:where rdf:resource="#P_V1"/>`
	`  <vein:isAtASpeedOf`
	`        rdf:datatype="http://www.w3.org/2001/XMLSchema#double">`
	`   13.4112e0</vein:isAtASpeedOf>`
	`  <vein:previousJunction rdf:resource="vein#n97"/>`
	`  <vein:nextJunction rdf:resource="vein#i1"/>`
	`  <vein:currentRoadElement rdf:resource="vein#r2"/>`
	`  <vein:nextRoadElement rdf:resource="vein#r3"/>`
	`</vein:Private_Vehicle>`

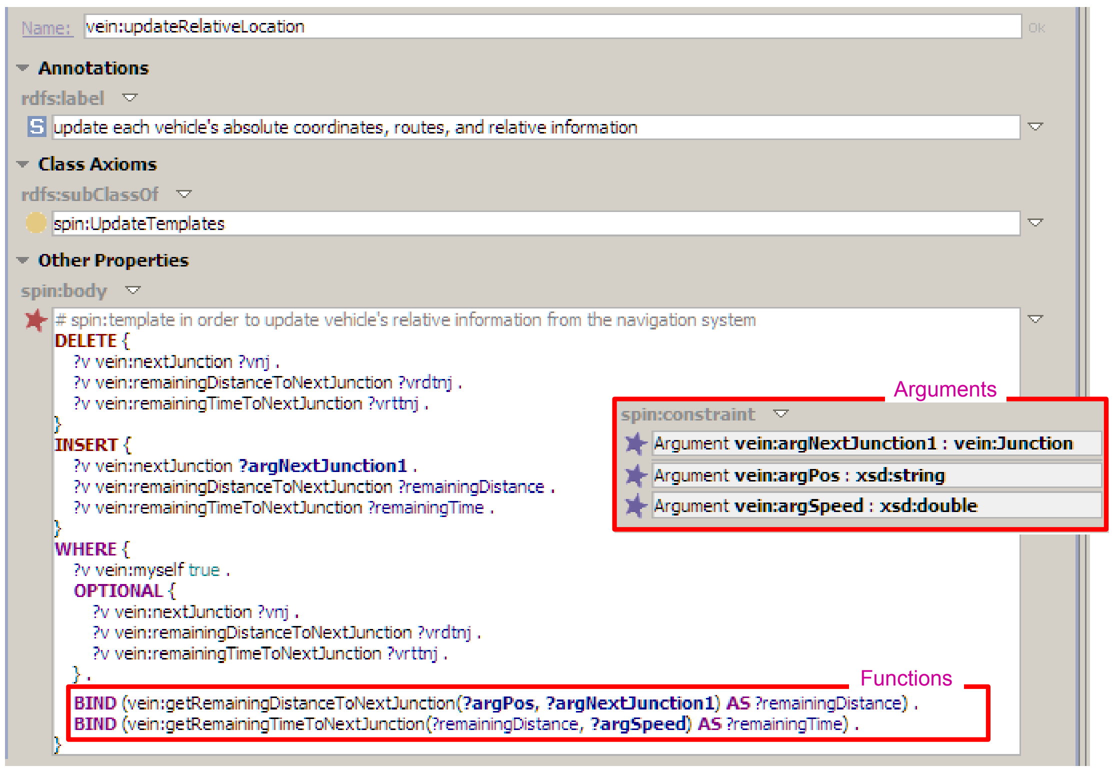
	`<gml:Point rdf:ID="P_V1">`
	`  <gml:pos`
	`          rdf:datatype="http://www.w3.org/2001/XMLSchema#string">`
	`   517426.918 169319.8</gml:pos>`
	`</gml:Point>`

Intersection i1's location	`<vein:Intersection rdf:about="vein#i1">`
	`  <geo:where rdf:resource="vein#P_i1"/>`
	`</vein:Intersection>`

	`<gml:Point rdf:about="vein#P_i1">`
	`  <gml:pos`
	`          rdf:datatype="http://www.w3.org/2001/XMLSchema#string">`
	`   517479.987 169349.316</gml:pos>`
	`</gml:Point>`

Here, it is assumed that a navigation system is implanted in each vehicle, and each vehicle's VEIN ontology model is integrated with its navigation system, so that partial route and relative location information of each vehicle's ontology model can be updated in the same cycle in which its absolute location and the turn indicator information are updated in the navigation system. In this circumstance, a vehicle's relative location can be inferred from existing asserted triples (e.g. its absolute location and partial route information) and SPIN rules.

However, the VEIN application ontology uses SPIN functions and templates rather than SPIN rules to generate the vehicle's relative location as an asserted triple, not as an inferred triple. Usually, keeping inferences up to date after each change of assertions will be efficient if the asserted data remain static and rarely change, but when dealing with fast-moving vehicles, it is more efficient and accurate to compute the values of interest dynamically on demand using SPIN functions (Knublauch, 2011). As obtaining relative information is a very repetitive process, a SPIN template is defined in the VEIN model to support this cyclic update of relative information (Figure 5.9). The SPIN query template uses three arguments (inputs) and two SPIN functions to obtain the vehicle's information (coming junction, current coordinates, and current speed) and to compute its relative information (remaining distance and time to the next junction), respectively.



Name: vein:updateRelativeLocation

rdfs:label: update each vehicle's absolute coordinates, routes, and relative information

rdfs:subClassOf: spin:UpdateTemplates

spin:body:

```
# spin:template in order to update vehicle's relative information from the navigation system
DELETE {
   ?v vein:nextJunction ?vnj .
   ?v vein:remainingDistanceToNextJunction ?vrdtnj .
   ?v vein:remainingTimeToNextJunction ?vrttnj .
}
INSERT {
   ?v vein:nextJunction ?argNextJunction1 .
   ?v vein:remainingDistanceToNextJunction ?remainingDistance .
   ?v vein:remainingTimeToNextJunction ?remainingTime .
}
WHERE {
   ?v vein:myself true .
   OPTIONAL {
      ?v vein:nextJunction ?vnj .
      ?v vein:remainingDistanceToNextJunction ?vrdtnj .
      ?v vein:remainingTimeToNextJunction ?vrttnj .
   } .
   BIND (vein:getRemainingDistanceToNextJunction(?argPos, ?argNextJunction1) AS ?remainingDistance) .
   BIND (vein:getRemainingTimeToNextJunction(?remainingDistance, ?argSpeed) AS ?remainingTime) .
}
```

spin:constraint:

```
Argument vein:argNextJunction1 : vein:Junction
Argument vein:argPos : xsd:string
Argument vein:argSpeed : xsd:double
```

Figure 5.9 - A SPIN template to update vehicle's relative location information

Recently, the stream reasoning approach that integrates data streams and reasoning systems is gaining popularity for supporting various traffic applications (e.g. traffic monitoring, traffic pattern detection). The applications deal with time-varying data elements, with a one-timestamp model (e.g. Continuous-SPARQL, SPARQLstream) or a two-timestamps model (e.g. EP-SPARQL/Etalis) (Valle et al., 2009; Valle, 2014). For example, in a traffic management system, a registered continuous query, which gets streams of answers from input streams, can be used to search slow traffic events and automatically modify a speed limit on a certain road section (Anicic et al., 2012). Thus, stream reasoning is necessary when dealing with incrementally available, multiple, heterogeneous, gigantic data streams 'to support the decision process of extremely large numbers of concurrent users' in real-time (Stuckenschmidt et al., 2010). However, continuous semantics are still challenging, and many relevant reasoning methods are not able to deal with high frequency data streams (Stuckenschmidt et al., 2010; Valle, 2014). This research deals with an ontology as a vehicular messaging model to support a local data transmission between a rapidly moving ambulance and nearby vehicles on top of DSRC. The vehicles can send/receive information to one another at an adjustable information exchange rate, in which maximum possible rate is approximately 10 times per second, i.e. about 100 ms. Therefore, the contents and the processing of the message have to be designed minimally to keep DSRC's low latency. Even in a real-time streaming processing system, live feeds provide a major input of data to maintain the system's current state by processing them in memory. For these reasons, storing or querying streaming data can be an optional process (Mladenić et al., 2012), and the stream data/reasoning approach is not considered in this research.

As we have seen so far, SPIN functions and templates can be useful to update a vehicle's relative location and route information. Each vehicle's relative information depends on the route information, which is determined by its origin and destination on the road network, so real-time traffic information, such as road restriction or road traffic status, can be crucial in reality. The road network information and traffic situation can be shared via vehicular communications, but this research deals with the physical road network only, with the assumption that the instances of road elements and junctions are simply stored as asserted triples in the VEIN ontology model. Consequently, each vehicle obtains the road network information from the VEIN model in order to compute the relative information.

The next section explains how the ambulance of the ambulance scenario and the broken-down car of the breakdown scenario can each generate/send a message (partial information) from its own ontology model and how recipient vehicles can instantiate the ambulance or broken-down car into their ontology model to calculate their relative locations and make decisions.

5.5 Validation of the VEIN application ontology

To demonstrate how the VEIN application ontology can be used in dynamic situations, two tasks are developed in a time sequence based on snapshots of the two scenarios (Figure 4.5 and Figure 4.9). Following the sequential tasks, vehicles and their movements and interactions are instantiated by using the classes, properties, and relations of the application ontology.

Section 5.5.1 describes vehicles' interactions in a time sequence as tasks to validate the VEIN application ontology. It shows two time sequences for the ambulance scenario and the breakdown vehicle scenario. Section 5.5.2 instantiates the ambulance and the vehicles around it in order to demonstrate how information about the ambulance's dynamic movements can be shared with the vehicles in the vicinity of the ambulance by updating their ontology model. Section 5.5.3 explains instances of the broken-down vehicle and approaching vehicles to the breakdown spot on the motorway. Since the main role of SPIN functions and templates is to update vehicles' ontology model for each snapshot of the scenarios, their logics and return values are also presented.

5.5.1 Sequential description of the interactions between vehicles

Since the VEIN application ontology has been developed for an ITS application based on scenarios, it can be evaluated by using it for specific tasks from the scenarios. In this section, two sequential interaction diagrams are used as tasks for the two scenarios.

First, sequential tasks of an ambulance and a private vehicle are used to evaluate the application ontology for the first scenario (Figure 5.10). The ambulance and the private vehicle obtain their absolute coordinates from a GPS receiver iteratively. When they obtain new location coordinates, they execute a SPIN query template to obtain relative locations. In the ambulance's communication message, its relative location will be included to share its presence and movement. When the ambulance broadcasts a communication message to vehicles around it and a private vehicle receives the message, the private vehicle executes a SPIN query template to create, update, or delete the ambulance's instance. When the private vehicle receives the message for the first time, the ambulance' instance is created in its ontology model. Then, whenever the private vehicle receives a new message from the ambulance, the ambulance's instance is updated in the private vehicle's ontology model to keep it up to date.

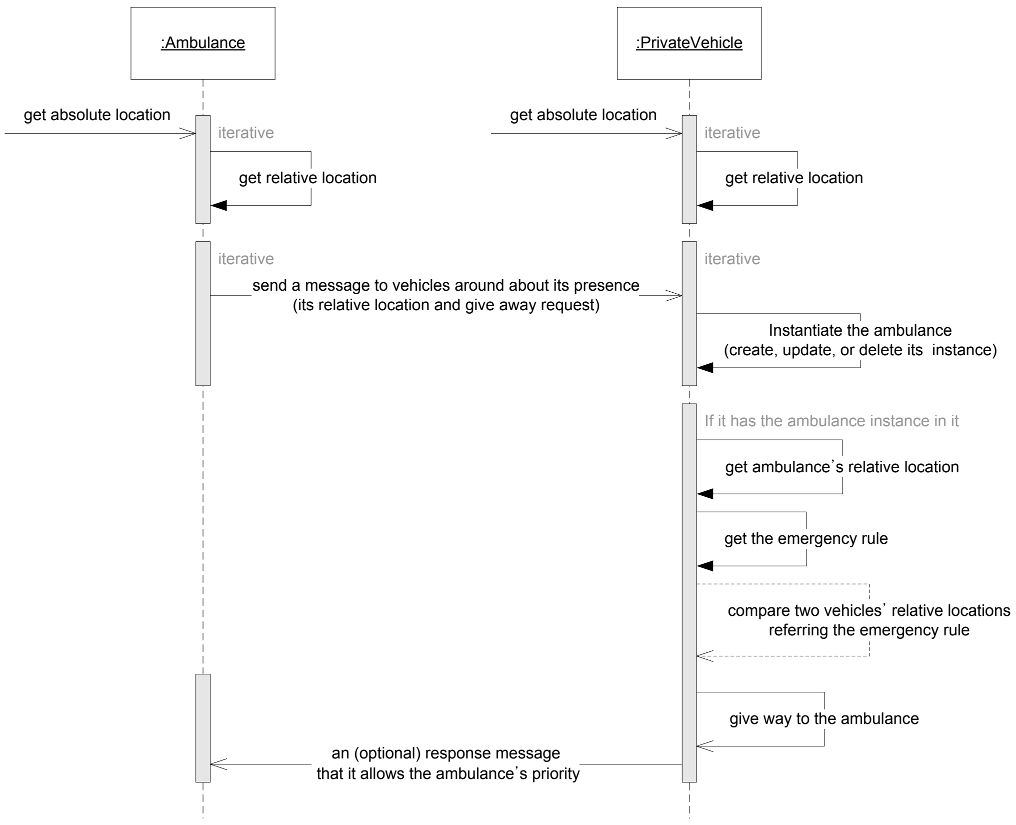
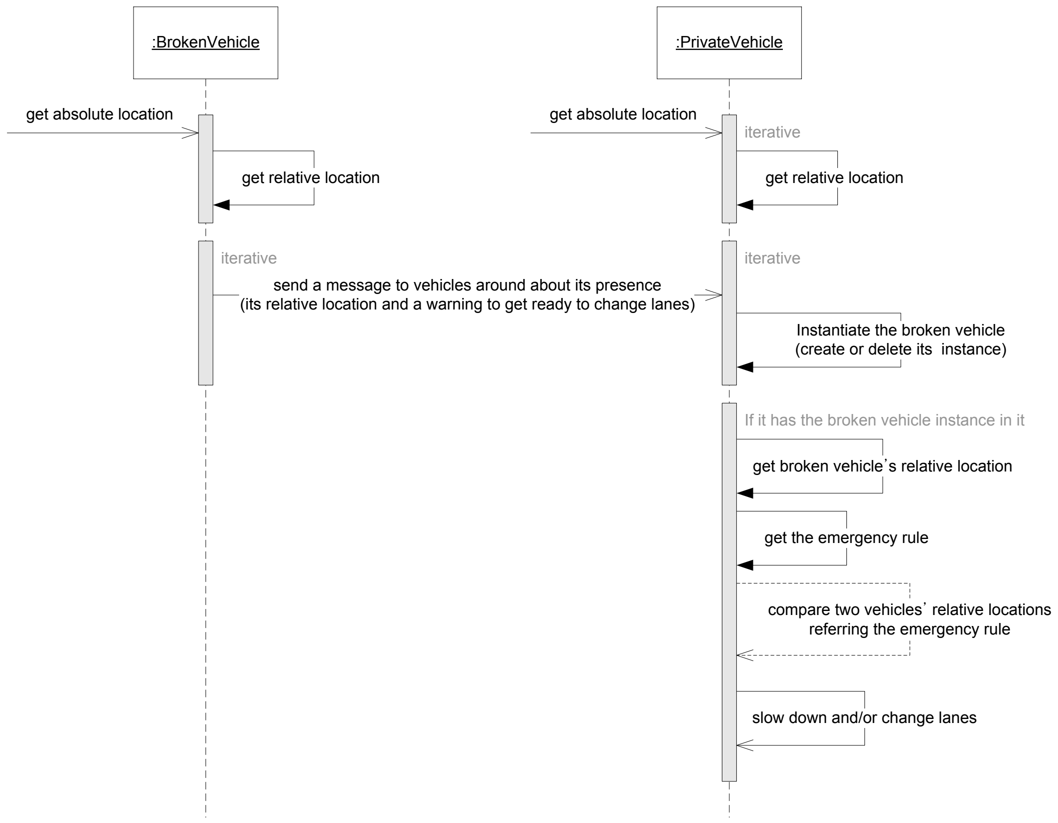

:Ambulance
:PrivateVehicle
get absolute location
iterative
get relative location
get absolute location
iterative
get relative location
iterative
send a message to vehicles around about its presence
(its relative location and give away request)
iterative
Instantiate the ambulance
(create, update, or delete its instance)
If it has the ambulance instance in it
get ambulance's relative location
get the emergency rule
compare two vehicles' relative locations
referring the emergency rule
give way to the ambulance
an (optional) response message
that it allows the ambulance's priority

Figure 5.10 - Sequential tasks of an ambulance and a private vehicle

The private vehicle has its own instance and the ambulance's instance in its ontology model, therefore it can obtain the ambulance's relative location and the emergency rule. The emergency rule can be passed from the ambulance, or just some parameters of the emergency rule can be included in the communication message if the ambulance's request is based on the emergency rule that is already stored in the VEIN ontology. Normally, the aim of the emergency rule is to assist vehicles potentially affected by the ambulance, and the emergency rule is also generated in the format of a SPIN template. So, the ambulance's relative location and the private vehicle's location are used as parameters of the SPIN template, and the logic of the emergency rule is stored in the template. If the private vehicle has the same route as the ambulance and they are close enough, the emergency rule template will return a value for giving way to the ambulance. The communication has to be minimised, but if it is worthwhile the private vehicle answering the ambulance to resolve the situation, it is possible to do so.

Second, the sequential tasks of a broken-down vehicle and a private vehicle are developed from the breakdown scenario to evaluate the application ontology (Figure 5.11). These sequential tasks are quite similar to the sequential tasks of the ambulance scenario, but there are differences. Since the broken-down vehicle is a stationary vehicle, there is no iterative process involved in obtaining its position. The message from the broken-down vehicle is also static, so there is no update for the broken-down vehicle instance in the ontology model of the private vehicle. In addition, the message from the broken-down vehicle is a warning message, so the private vehicle may not answer the broken-down vehicle after it has slowed down and changed lanes.



:BrokenVehicle
:PrivateVehicle
get absolute location
get relative location
get absolute location
iterative
get relative location
iterative
iterative
send a message to vehicles around about its presence (its relative location and a warning to get ready to change lanes)
Instantiate the broken vehicle (create or delete its instance)
If it has the broken vehicle instance in it
get broken vehicle's relative location
get the emergency rule
compare two vehicles' relative locations referring the emergency rule
slow down and/or change lanes

Figure 5.11 - Sequential tasks for a broken-down vehicle and a private vehicle

Here, we described the two sequential tasks representing the interactions of the two scenarios to evaluate the application ontology. The sequential tasks describe subtasks necessary for a vehicle that receives a message from an ambulance or a broken-down vehicle to perform an action (e.g. give way to the ambulance, slow down, and change lanes). The following two subsections describe vehicle instances to demonstrate how vehicles can perform the tasks with the application ontology for each scenario.

5.5.2 Instantiating vehicles and their interactions for the ambulance scenario

Interactions between the ambulance and the vehicle, as shown in the ambulance scenario of Section 4.3.1 and in Figure 5.10, are implemented based on the VEIN ontology model and DSRC messages. Each vehicle's ontology model inherits the VEIN model for the road network information, classes and properties for vehicles, and SPIN functions and templates. As shown in Figure 5.12, each vehicle's ontology model has its own instance to represent its absolute and relative location, which can be updated via its navigation system and SPIN templates. When a vehicle `V1` receives DSRC messages from an ambulance `A1`, the ontology model of the vehicle uses the DSRC messages as inputs of a query template repeatedly to instantiate the ambulance in it. Focusing on the ontology model, it is assumed that there is a pseudo DSRC message set that broadcasts the relative locations of the ambulance to support the decision-making regarding whether the vehicle needs to give way to the ambulance.

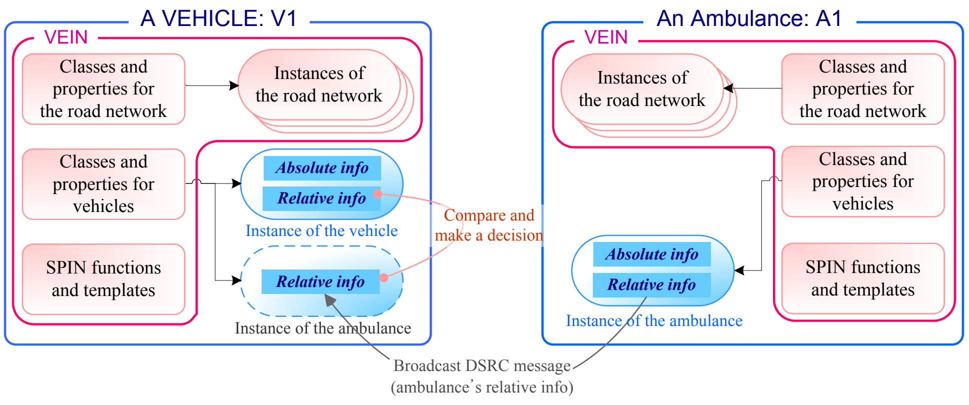
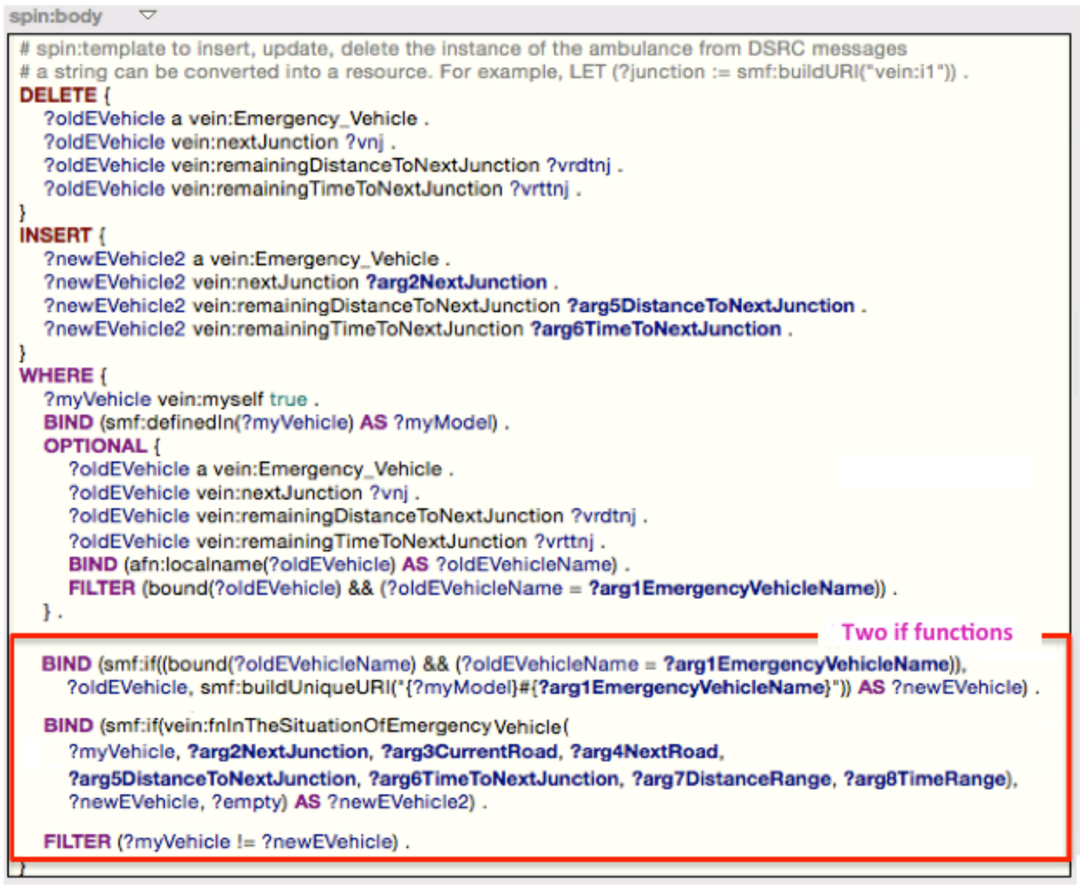

A VEHICLE: V1
VEIN
Classes and properties for the road network
Instances of the road network
Classes and properties for vehicles
Absolute info
Relative info
Instance of the vehicle
SPIN functions and templates
Relative info
Instance of the ambulance
Compare and make a decision
An Ambulance: A1
VEIN
Instances of the road network
Classes and properties for the road network
Classes and properties for vehicles
Absolute info
Relative info
Instance of the ambulance
SPIN functions and templates
Broadcast DSRC message (ambulance's relative info)

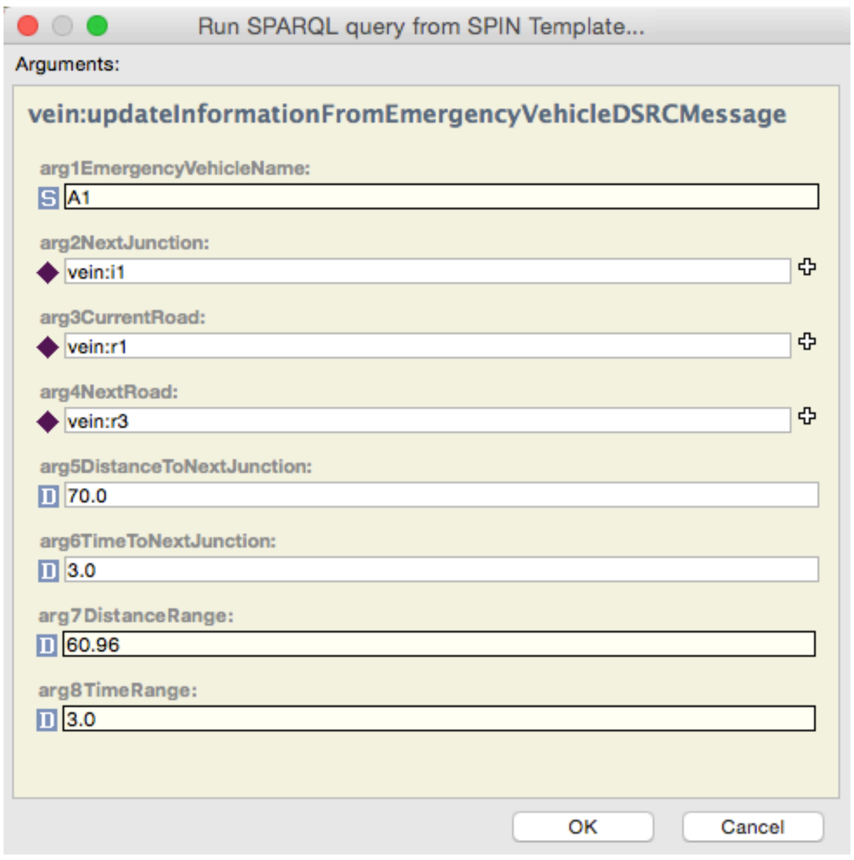
Figure 5.12 - Interaction between an ambulance and a vehicle based on the ontology model and DSRC message

When the vehicle receives a DSRC message from the ambulance, the vehicle can use the DSRC message to update its own ontology model by executing an update query from the SPIN update template (i.e. `vein:updateInformationFromEmergencyVehicleDSRCMessage`) (Figure 5.13). When the vehicle receives the message from the ambulance, an update query based on the query template can be executed to create, update, or delete the instance of the ambulance in its own ontology model. To instantiate the ambulance in the vehicle's ontology model, the query template uses several variables, such as `oldEVehicleAmbul`, `newEVehicleAmbul`, and `newEVehicleAmbul2`, to check if the ambulance's instance has already been created in its ontology model and, if the vehicle is in the ambulance' vicinity, to give way to the ambulance. If the `oldEVehicle` variable's value is null, the `newEVehicle` variable's value is not null, and the `newEVehicle2` variable's value is null, the vehicle does not have the ambulance's instance in its ontology model because the vehicle is not yet in the effect zone of the ambulance. If the `oldEVehicle` variable's value is not null and the `newEVehicle2` variable's value is null, the query template deletes the ambulance's instance from the vehicle's ontology model because the vehicle is no longer in the effect zone of the ambulance. Figure 5.14 shows how the SPIN template can use DSRC messages as its arguments to execute an update query for the vehicle's ontology model. The SPIN template uses eight arguments based on the DSRC message from the ambulance, such as the ambulance's unique name, its relative location, and the guided distance and time to request its priority (i.e. 200 meters and 10 seconds in this case). Using these arguments, the SPIN update template compares partial routes and relative locations between the ambulance and the vehicle, and supports the vehicle's action to give way to the ambulance.



```
# spin:template to insert, update, delete the instance of the ambulance from DSRC messages
# a string can be converted into a resource. For example, LET (?junction := smf:buildURI("vein:i1")) .
DELETE {
   ?oldEVehicle a vein:Emergency_Vehicle .
   ?oldEVehicle vein:nextJunction ?vnj .
   ?oldEVehicle vein:remainingDistanceToNextJunction ?vrdtnj .
   ?oldEVehicle vein:remainingTimeToNextJunction ?vrttnj .
}
INSERT {
   ?newEVehicle2 a vein:Emergency_Vehicle .
   ?newEVehicle2 vein:nextJunction ?arg2NextJunction .
   ?newEVehicle2 vein:remainingDistanceToNextJunction ?arg5DistanceToNextJunction .
   ?newEVehicle2 vein:remainingTimeToNextJunction ?arg6TimeToNextJunction .
}
WHERE {
   ?myVehicle vein:myself true .
   BIND (smf:definedIn(?myVehicle) AS ?myModel) .
   OPTIONAL {
      ?oldEVehicle a vein:Emergency_Vehicle .
      ?oldEVehicle vein:nextJunction ?vnj .
      ?oldEVehicle vein:remainingDistanceToNextJunction ?vrdtnj .
      ?oldEVehicle vein:remainingTimeToNextJunction ?vrttnj .
      BIND (afn:localname(?oldEVehicle) AS ?oldEVehicleName) .
      FILTER (bound(?oldEVehicle) && (?oldEVehicleName = ?arg1EmergencyVehicleName)) .
   } .

   BIND (smf:if((bound(?oldEVehicleName) && (?oldEVehicleName = ?arg1EmergencyVehicleName)),
      ?oldEVehicle, smf:buildUniqueURI("{?myModel}#{?arg1EmergencyVehicleName}")) AS ?newEVehicle) .

   BIND (smf:if(vein:fnInTheSituationOfEmergencyVehicle(
      ?myVehicle, ?arg2NextJunction, ?arg3CurrentRoad, ?arg4NextRoad,
      ?arg5DistanceToNextJunction, ?arg6TimeToNextJunction, ?arg7DistanceRange, ?arg8TimeRange),
      ?newEVehicle, ?empty) AS ?newEVehicle2) .

   FILTER (?myVehicle != ?newEVehicle) .
}
```

Figure 5.13 - The SPIN template to instantiate the ambulance in each vehicle's ontology model



vein:updateInformationFromEmergencyVehicleDSRCMessage

arg1EmergencyVehicleName: A1

arg2NextJunction: vein:i1

arg3CurrentRoad: vein:r1

arg4NextRoad: vein:r3

arg5DistanceToNextJunction: 70.0

arg6TimeToNextJunction: 3.0

arg7DistanceRange: 60.96

arg8TimeRange: 3.0

Figure 5.14 - Argument values[22] of the SPIN update template from the ambulance's DSRC message

The SPIN template also uses two IF functions inside the BIND clauses to set variables based on the return value of some logics, as highlighted in the lower rectangle of Figure 5.13. The first IF function of the SPIN template checks if an instance of the ambulance exists in the vehicle's ontology model. If an instance of the ambulance already exists in the vehicle's ontology model, the IF function returns 'true'. The second IF function of the template checks whether the vehicle needs to give way to the ambulance based on a decision-making logic (i.e. emergency rules)..

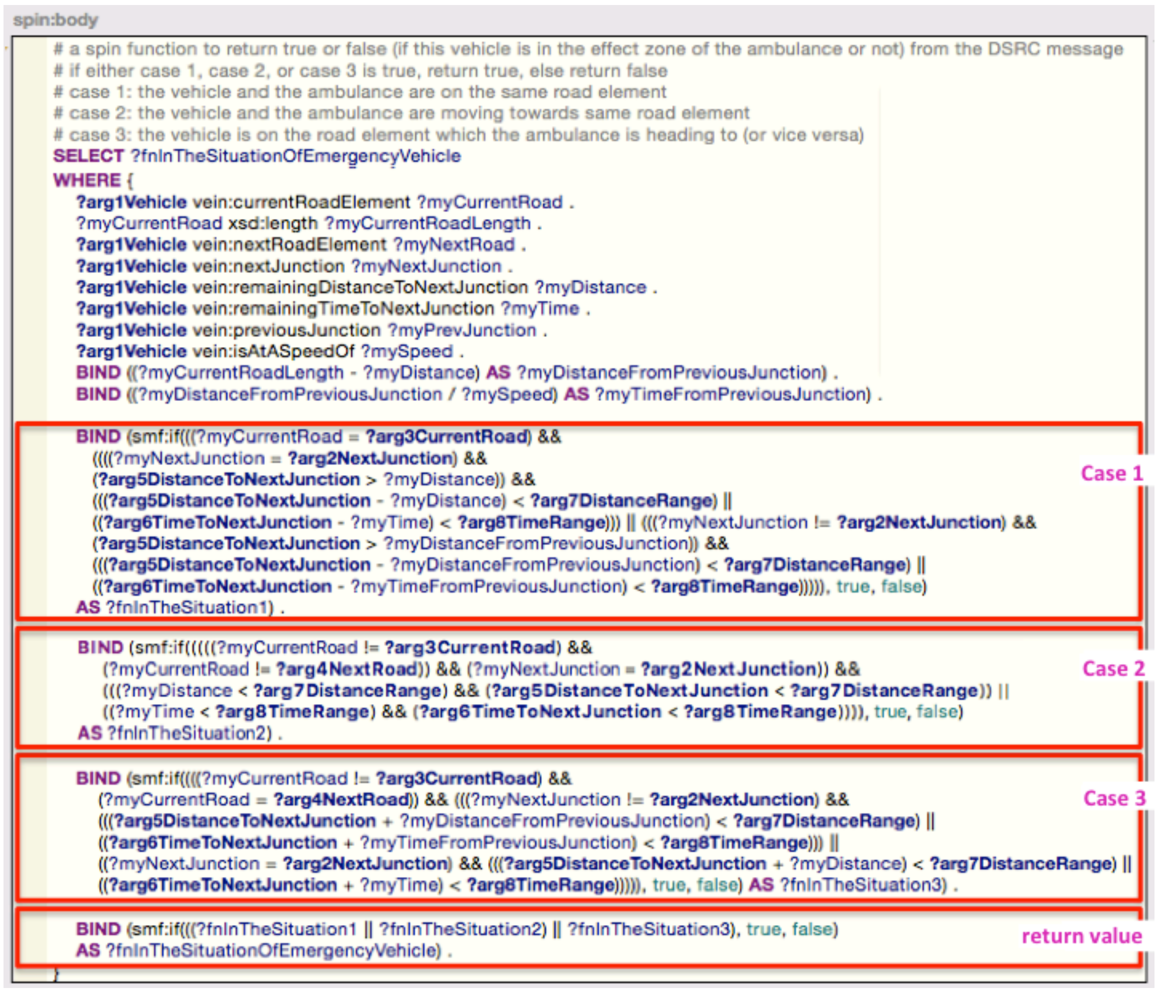
To develop the decision-making logic, and the emergency rules of the SPIN function (`fnInTheSituationOfEmergencyVehicle`) that contain three different cases are explained with pseudo code and descriptions (Table 5.3). The emergency rules are then

[22] Note that the second, third, and fourth arguments are regarded as objects for the sake of simplicity. However, in a real communication, these objects' unique identifiers can be used instead to minimise the volume of the sender's DSRC message set. The argument for the given distance range (arg7DistanceRange) was set to 60.96 meters by referring to the effect range of the new howler sirens of ambulances in Oklahoma, U.S.A. According to the Emergency Medical Services Authority (EMSA, 2009) in Oklahoma, the howler sirens emit low-frequency tones that cause objects within 200 feet (i.e. 60.96 metres) to reverberate, and catch drivers' attention much more quickly. The argument for the given time range was set to 3.0sec (60.96 m / (30mph *1.5)).

implemented as a SPIN function (i.e. `vein:fnInTheSituationOfEmergencyVehicle` in Figure 5.15). In the first case, the vehicle and the ambulance are on the same road element, the vehicle is in front of the ambulance. Their distance (or time) is shorter than the distance (or time) range of the ambulance's request, which are 7th and 8th arguments of the SPIN update template as parts of the ambulance's DSRC message. It means that if the vehicle is within the request zone of the ambulance, it needs to give way to the ambulance. The second case shows that the vehicle and the ambulance are not on the same road element, the vehicle is not on the road element which the ambulance is heading towards, they are moving towards the same junction, and their distance (or time) is shorter than the given distance/time range. Lastly, in the third case, the vehicle is on the road element which the ambulance is moving towards, the vehicle is in front of the ambulance, and their distance (or time) is shorter than the given range.

Table 5.3 - Logics of the function to check whether or not the vehicle needs to give way to the ambulance

Situation	Pseudo Code	Comments
Case 1	If (V1's current road == A1's current road AND V1's next junction == A1's next junction AND (A1's distance to next junction > V1's distance to next junction) AND (A1's distance/time to next junction - V1's distance/time to next junction < given distance/time range)) then V1 is in the situation of Case 1	• V1 and A1 are on the same road • They are moving in the same direction • V1 is in front of A1 • Their distance (or time) is shorter than given distance range (or time range) • Give-way request and Forward collision warning
	If (V1's current road == A1's current road AND V1's next junction != A1's next junction AND (A1's distance to next junction > V1's distance from previous junction) AND (A1's distance/time to next junction - V1's distance/time from previous junction < given distance/time)) then V1 is in the situation of Case 1	• V1 and A1 are on the same road • They are moving in the opposite direction • V1 is in front of A1 • Their distance (or time) is shorter than given distance range (or time range) • Give-way request and Head-on collision warning
Case 2	If (V1's current road :isConnectedTo A1's current road AND V1's current road :isConnectedTo A1's next road AND V1's next junction == A1's next junction AND ((A1's distance to next junction < given distance range AND V1's distance to next junction < given distance range OR (A1's time to next junction < given time range AND V1's time to next junction < given time range))) then V1 is in the situation of Case 2	• V1 and A1 are not on the same road • V1 is not on the road that A1 is heading to • V1 is on the road that is connected to A1's current road and A1's next road • V1 and A1 are coming to the same junction • A1's distance (or time) to the junction is shorter than given distance range (or time range) • V1's distance (or time) to the junction is shorter than chosen suggested giving-way distance range (or time range) • Possible suggested giving-way ranges - given distance range (or time range) like other cases - (2 * A1' distance (or time) to the junction + given distance range (or time range)) / 3 - A1's distance (or time) to the junction - Two-second gap to the junction • Give-way request and Intersection collision warning
Case 3	If (V1's current road != A1's current road AND V1's current road == A1's next road AND V1's next junction != A1's next junction AND (A1's distance/time to next junction + V1's distance/time from previous junction < given distance/time range)) then V1 is in the situation of Case 3	• V1 and A1 are not on the same road • V1 is on the road that A1 is heading to • V1 is in front of A1 • They are moving in the same direction • Their distance (or time) is shorter than given distance (or time) • Give-way request and Forward collision warning
	If (V1's current road != A1's current road AND V1's current road == A1's next road AND V1's next junction == A1's next junction AND (A1's distance/time to next junction + V1's distance/time to next junction < given distance/time range)) then V1 is in the situation of Case 3	• V1 and A1 are not on the same road • V1 is on the road that A1 is heading to • V1 is in front of A1 • They are moving in the opposite direction • Their distance (or time) is shorter than given distance (or time) • Give-way request and Head-on collision warning



```
# a spin function to return true or false (if this vehicle is in the effect zone of the ambulance or not) from the DSRC message
# if either case 1, case 2, or case 3 is true, return true, else return false
# case 1: the vehicle and the ambulance are on the same road element
# case 2: the vehicle and the ambulance are moving towards same road element
# case 3: the vehicle is on the road element which the ambulance is heading to (or vice versa)
SELECT ?fnInTheSituationOfEmergencyVehicle
WHERE {
   ?arg1Vehicle vein:currentRoadElement ?myCurrentRoad .
   ?myCurrentRoad xsd:length ?myCurrentRoadLength .
   ?arg1Vehicle vein:nextRoadElement ?myNextRoad .
   ?arg1Vehicle vein:nextJunction ?myNextJunction .
   ?arg1Vehicle vein:remainingDistanceToNextJunction ?myDistance .
   ?arg1Vehicle vein:remainingTimeToNextJunction ?myTime .
   ?arg1Vehicle vein:previousJunction ?myPrevJunction .
   ?arg1Vehicle vein:isAtASpeedOf ?mySpeed .
   BIND ((?myCurrentRoadLength - ?myDistance) AS ?myDistanceFromPreviousJunction) .
   BIND ((?myDistanceFromPreviousJunction / ?mySpeed) AS ?myTimeFromPreviousJunction) .

   BIND (smf:if(((?myCurrentRoad = ?arg3CurrentRoad) &&
     ((((?myNextJunction = ?arg2NextJunction) &&
     (?arg5DistanceToNextJunction > ?myDistance)) &&
     (((?arg5DistanceToNextJunction - ?myDistance) < ?arg7DistanceRange) ||
     ((?arg6TimeToNextJunction - ?myTime) < ?arg8TimeRange))) || (((?myNextJunction != ?arg2NextJunction) &&
     (?arg5DistanceToNextJunction > ?myDistanceFromPreviousJunction)) &&
     (((?arg5DistanceToNextJunction - ?myDistanceFromPreviousJunction) < ?arg7DistanceRange) ||
     ((?arg6TimeToNextJunction - ?myTimeFromPreviousJunction) < ?arg8TimeRange))))), true, false)
   AS ?fnInTheSituation1) .

   BIND (smf:if(((((?myCurrentRoad != ?arg3CurrentRoad) &&
      (?myCurrentRoad != ?arg4NextRoad)) && (?myNextJunction = ?arg2NextJunction)) &&
      (((?myDistance < ?arg7DistanceRange) && (?arg5DistanceToNextJunction < ?arg7DistanceRange)) ||
      ((?myTime < ?arg8TimeRange) && (?arg6TimeToNextJunction < ?arg8TimeRange)))), true, false)
   AS ?fnInTheSituation2) .

   BIND (smf:if((((?myCurrentRoad != ?arg3CurrentRoad) &&
     (?myCurrentRoad = ?arg4NextRoad)) && (((?myNextJunction != ?arg2NextJunction) &&
     (((?arg5DistanceToNextJunction + ?myDistanceFromPreviousJunction) < ?arg7DistanceRange) ||
     ((?arg6TimeToNextJunction + ?myTimeFromPreviousJunction) < ?arg8TimeRange))) ||
     ((?myNextJunction = ?arg2NextJunction) && (((?arg5DistanceToNextJunction + ?myDistance) < ?arg7DistanceRange) ||
     ((?arg6TimeToNextJunction + ?myTime) < ?arg8TimeRange))))), true, false) AS ?fnInTheSituation3) .

   BIND (smf:if(((?fnInTheSituation1 || ?fnInTheSituation2) || ?fnInTheSituation3), true, false)
   AS ?fnInTheSituationOfEmergencyVehicle) .
}
```

Figure 5.15 - The SPIN function to check if a vehicle needs to give way to the ambulance or not

When the vehicle receives a message from the ambulance, the update query for the ambulance's instance is executed, and the function (`fnInTheSituationOfEmergencyVehicle`) inside the update query template returns 'true' only if the vehicle is in one of these three cases. In the `fnInTheSituationOfEmergencyVehicle` function, the `remainingTimeToNextJunction` [23] property is used as a subsidiary to the `remainingDistanceToNextJunction` property, so these two properties are used as one

[23] The `remainingTimeToNextJunction` property cannot be used alone as a replacement of the `remainingDistanceToNextJunction` property. There can be a situation in which a vehicle is heading towards an ambulance and it is moving extremely fast, so that the defined distance range may not give the vehicle enough time to react. With the defined time range, this property is devised to support this kind of situation that is beyond the design of the defined distance range. For example, a vehicle is heading to an ambulance and it is moving extremely fast, even though the vehicle is outside the defined distance range the vehicle will be requested to give way to the ambulance if it is in the defined time range.

set. Table 5.4 summarises the return values of the functions and the results of the update query template for each time stamp of the ambulance scenario, which is described in Figure 4.5.

Table 5.4 - Results of SPIN functions and the SPIN template for each time stamp

Time Stamp	V1's relative location	A1's relative location	Return value of the function A*	Return value of the function B**	Working clause of the Update Query Template
Figure 4.5 a	on r2, to i1, 60 meters, 4.5 seconds	on r1, to i1, 70 meters, 3.0 seconds	False	True	only INSERT clause
Figure 4.5 b	on r2, to i1, 20 meters, 3 seconds	on r1, to i1, 35 meters, 1.5 seconds	True	True	DELETE clause and INSERT clause
Figure 4.5 c	on r3, to n1, 60 meters, 4.5 seconds	on r3, to n1, 50 meters, 2.1 seconds	True	False	only DELETE clause

* Function A is the first of the two if functions (red boxed in Figure 5.13). It checks if the ambulance's instance exists already in the vehicle's ontology model.
** Function B is the second of the two if functions (red boxed in Figure 5.13). It checks if the vehicle is inside the effect zone of the ambulance (and needs to give way to the ambulance).

5.5.3 Instantiating vehicles and their interactions for the breakdown scenario

Interactions in the breakdown scenario of Section 4.3.2 and sequential tasks in Figure 5.11, which is a breakdown situation on the motorway, are also implemented based on the VEIN application ontology and DSRC messages. Each vehicle's VEIN ontology model and the updating process is the same as described in Figure 5.12, but the communication subject broadcasting an alert message is the broken-down vehicle instead of the ambulance (Figure 5.16). When a vehicle (`V1`) in the vicinity of the breakdown spot receives DSRC messages from the broken-down vehicle (`B1`), the ontology model of the approaching vehicle uses the received messages as inputs of a query template repeatedly to instantiate the broken-down vehicle in it. Focusing on the ontology model, it is assumed that there is a pseudo DSRC message set that broadcasts the relative locations of the broken-down vehicle to warn of its presence on the motorway and to allow the vehicles approaching the breakdown spot to have a longer response time to avoid sudden braking or lane changes.

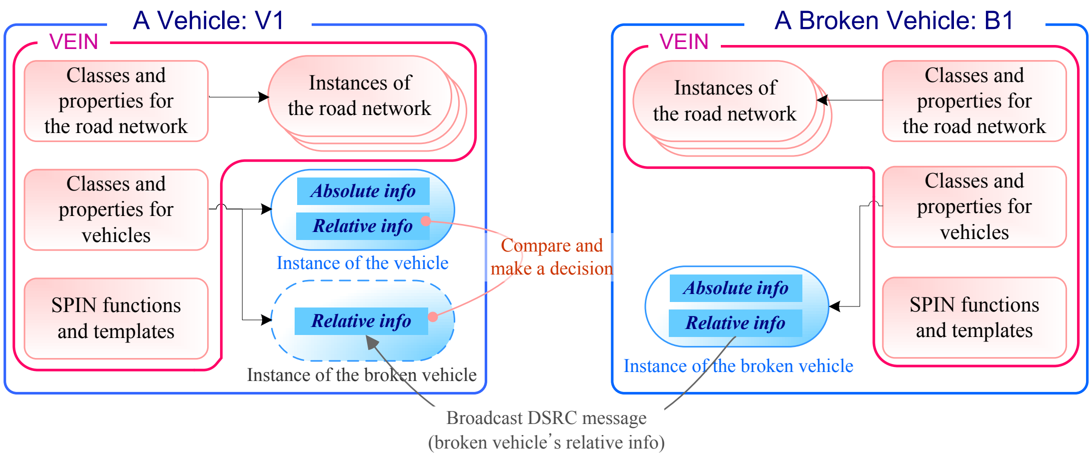
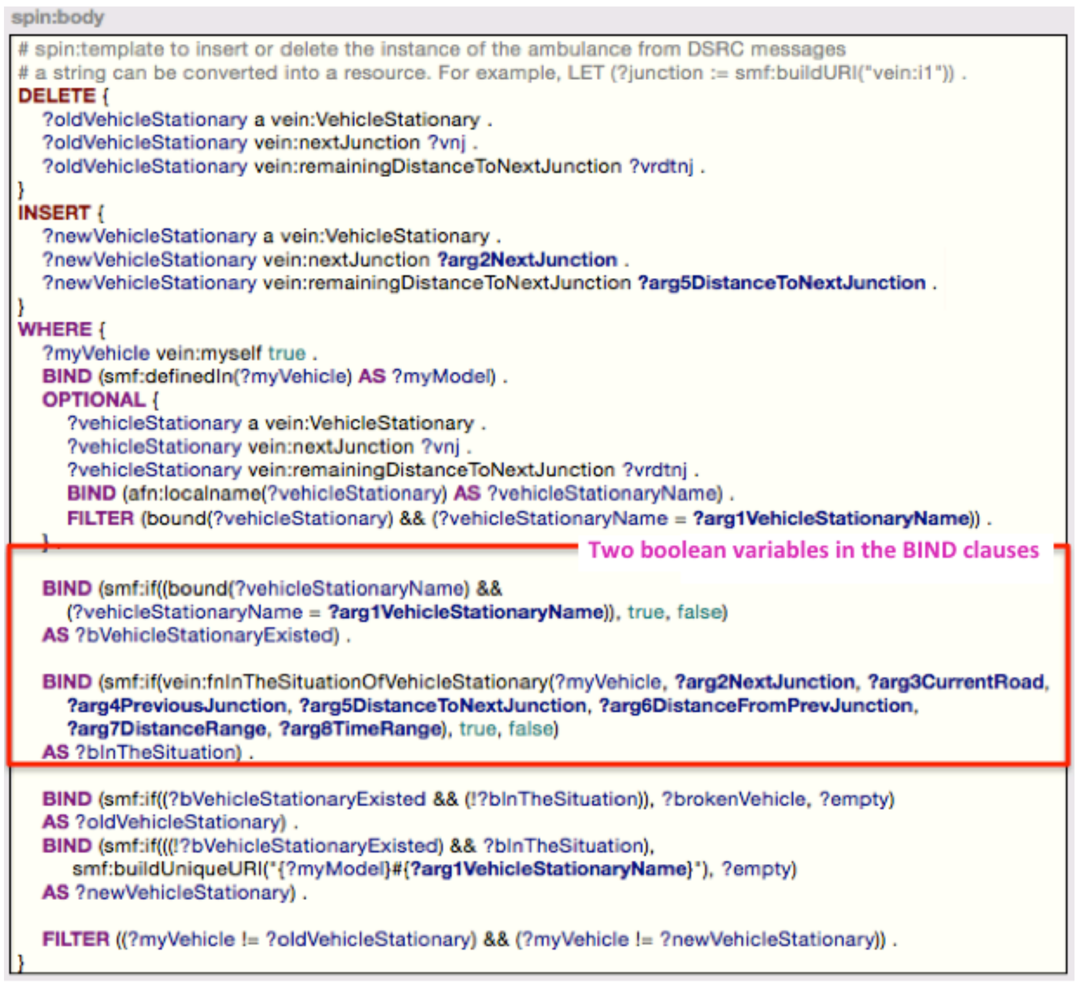

A Vehicle: V1
VEIN
Classes and properties for the road network
Instances of the road network
Classes and properties for vehicles
Absolute info
Relative info
Instance of the vehicle
SPIN functions and templates
Relative info
Instance of the broken vehicle
Compare and make a decision
A Broken Vehicle: B1
VEIN
Instances of the road network
Classes and properties for the road network
Classes and properties for vehicles
Absolute info
Relative info
Instance of the broken vehicle
SPIN functions and templates
Broadcast DSRC message (broken vehicle's relative info)

Figure 5.16 - Interaction between a broken-down vehicle and a vehicle approaching the breakdown spot based on the ontology model and DSRC message

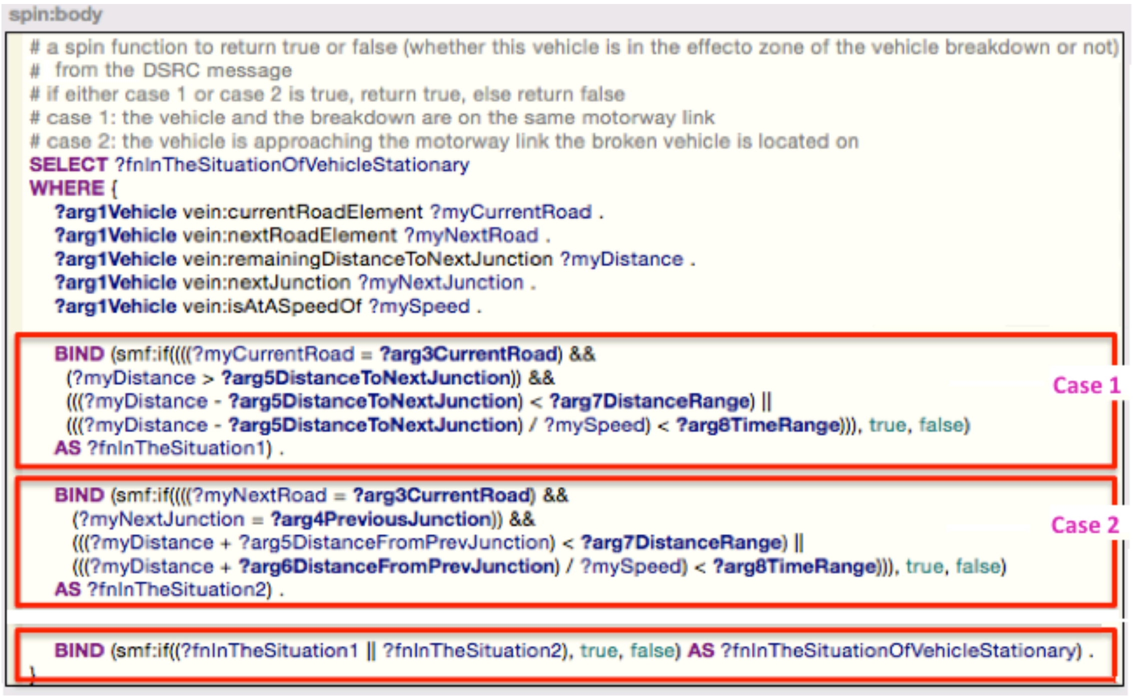
When the vehicle (`V1`) receives a DSRC message from the broken-down vehicle (`B1`), an update query can be executed to update the vehicle `V1`'s ontology model from the SPIN template (i.e. `vein:updateInformationFromVehicleStationaryDSRCMessage`) (Figure 5.17). As the broken-down vehicle is a stationary vehicle in the breakdown situation, the query template does not update the broken-down vehicle's location. In this situation, the DSRC message is required to inform about the presence of the vehicle breakdown, and the update query only creates or deletes the instance of the broken-down vehicle in the vehicle's ontology model based on the approaching vehicle's relative distance/time to the breakdown scene. The query template uses two boolean variables, which are `bVehicleStationaryExisted` and `bInTheSituation`, to check if the vehicle's ontology model contains the instance of the broken-down vehicle already and if the vehicle is in the effect zone of the vehicle breakdown (i.e. 1000 meters and 30 seconds in this case), respectively. If the instance of the breakdown already exists in the vehicle's ontology model, and the vehicle is no longer in the effect zone of the vehicle breakdown, the query template deletes the broken-down vehicle's instance. If the vehicle does not have the instance of the broken-down vehicle in its ontology model, and it is in the effect zone of the vehicle breakdown, the query template creates the instance of the broken-down vehicle in its own ontology model to warn about the presence of the vehicle breakdown.



```
# spin:template to insert or delete the instance of the ambulance from DSRC messages
# a string can be converted into a resource. For example, LET (?junction := smf:buildURI("vein:i1")) .
DELETE {
   ?oldVehicleStationary a vein:VehicleStationary .
   ?oldVehicleStationary vein:nextJunction ?vnj .
   ?oldVehicleStationary vein:remainingDistanceToNextJunction ?vrdtnj .
}
INSERT {
   ?newVehicleStationary a vein:VehicleStationary .
   ?newVehicleStationary vein:nextJunction ?arg2NextJunction .
   ?newVehicleStationary vein:remainingDistanceToNextJunction ?arg5DistanceToNextJunction .
}
WHERE {
   ?myVehicle vein:myself true .
   BIND (smf:definedIn(?myVehicle) AS ?myModel) .
   OPTIONAL {
      ?vehicleStationary a vein:VehicleStationary .
      ?vehicleStationary vein:nextJunction ?vnj .
      ?vehicleStationary vein:remainingDistanceToNextJunction ?vrdtnj .
      BIND (afn:localname(?vehicleStationary) AS ?vehicleStationaryName) .
      FILTER (bound(?vehicleStationary) && (?vehicleStationaryName = ?arg1VehicleStationaryName)) .
   } .

   BIND (smf:if((bound(?vehicleStationaryName) &&
      (?vehicleStationaryName = ?arg1VehicleStationaryName)), true, false)
   AS ?bVehicleStationaryExisted) .

   BIND (smf:if(vein:fnInTheSituationOfVehicleStationary(?myVehicle, ?arg2NextJunction, ?arg3CurrentRoad,
      ?arg4PreviousJunction, ?arg5DistanceToNextJunction, ?arg6DistanceFromPrevJunction,
      ?arg7DistanceRange, ?arg8TimeRange), true, false)
   AS ?bInTheSituation) .

   BIND (smf:if((?bVehicleStationaryExisted && (!?bInTheSituation)), ?brokenVehicle, ?empty)
   AS ?oldVehicleStationary) .
   BIND (smf:if(((!?bVehicleStationaryExisted) && ?bInTheSituation),
      smf:buildUniqueURI("{?myModel}#{?arg1VehicleStationaryName}"), ?empty)
   AS ?newVehicleStationary) .

   FILTER ((?myVehicle != ?oldVehicleStationary) && (?myVehicle != ?newVehicleStationary)) .
}
```

Figure 5.17 - The SPIN template to instantiate the broken-down vehicle in the approaching vehicles' ontology model

As shown in the red box of Figure 5.17, this SPIN query template uses a different SPIN function (`vein:fnInTheSituationOfVehicleStationary`) to check the situation of the vehicle breakdown ahead. This function only returns 'true' when the vehicle is in the two cases that can be defined by the vehicles' relative location with reference to motorway links and junctions. Two different cases of the function are described in Figure 5.18 and Table 5.5. In the first case, the broken-down vehicle and an approaching vehicle are on the same motorway link, and distance and time differences between two vehicles are shorter than the distance/time range the broken-down vehicle provides. In the second case, even though the broken-down vehicle and the approaching vehicle are not located on the same motorway link, the vehicle is moving towards the motorway link that the broken-down vehicle is located on, and their distance and time are shorter than the defined distance/time range. If the return value in either case is true, it means that the vehicle is within the guided distance and time to change the lanes to avoid the broken-down vehicle ahead.



```
# a spin function to return true or false (whether this vehicle is in the effecto zone of the vehicle breakdown or not)
#  from the DSRC message
# if either case 1 or case 2 is true, return true, else return false
# case 1: the vehicle and the breakdown are on the same motorway link
# case 2: the vehicle is approaching the motorway link the broken vehicle is located on
SELECT ?fnInTheSituationOfVehicleStationary
WHERE {
   ?arg1Vehicle vein:currentRoadElement ?myCurrentRoad .
   ?arg1Vehicle vein:nextRoadElement ?myNextRoad .
   ?arg1Vehicle vein:remainingDistanceToNextJunction ?myDistance .
   ?arg1Vehicle vein:nextJunction ?myNextJunction .
   ?arg1Vehicle vein:isAtASpeedOf ?mySpeed .

   BIND (smf:if((((?myCurrentRoad = ?arg3CurrentRoad) &&
     (?myDistance > ?arg5DistanceToNextJunction)) &&
     (((?myDistance - ?arg5DistanceToNextJunction) < ?arg7DistanceRange) ||
     (((?myDistance - ?arg5DistanceToNextJunction) / ?mySpeed) < ?arg8TimeRange))), true, false)
   AS ?fnInTheSituation1) .

   BIND (smf:if((((?myNextRoad = ?arg3CurrentRoad) &&
      (?myNextJunction = ?arg4PreviousJunction)) &&
      (((?myDistance + ?arg5DistanceFromPrevJunction) < ?arg7DistanceRange) ||
      (((?myDistance + ?arg6DistanceFromPrevJunction) / ?mySpeed) < ?arg8TimeRange))), true, false)
   AS ?fnInTheSituation2) .

   BIND (smf:if((?fnInTheSituation1 || ?fnInTheSituation2), true, false) AS ?fnInTheSituationOfVehicleStationary) .
}
```

Figure 5.18 - The SPIN function to check if a vehicle is inside the effect zone of the vehicle breakdown

Table 5.5 - Logics of the function to check whether or not a vehicle needs to consider the presence of the breakdown spot

Situation	Pseudo code	Comments
Case 1	If (V1's current motorway link == B1's current motorway link AND (V1's distance to next motorway junction > B1's distance to next motorway junction) AND (V1's distance to next motorway junction - B1's distance to next motorway junction < given distance range OR V1's travel time to B1's location < given time range)) then V1 is in the situation of Case 1	• V1 and B1 are on the same motorway link • B1 is in front of V1 • Their distance (or time) is shorter than given distance range (or time range)
Case 2	If (V1's next motorway link == B1's current motorway link AND (V1's distance to its next motorway junction + B1's distance from its previous motorway junction < given distance range OR V1's time to B1's location < given time range)) then V1 is in the situation of Case 2	• V1 and B1 are not on the same road link, but B1 is located on the road link that V1 is heading to • Their distance (or time) is less than given distance (or time)

Figure 4.9 shows a situation of a broken-down vehicle and a vehicle on a motorway link of M25 heading to Junction 11, the relative positions of the two vehicles are given by the broken-

vehicle `B1` and the vehicle `V2`. Here, the return values of the two boolean variables (`bVehicleStationaryExisted` and `bInTheSituation`) and the results of the update query template for each time stamp can be gained as shown in Table 5.6. If the vehicle `V2` has an instance of `B1` in its ontology model, it means that the vehicle `V2` is inside the effect zone of the vehicle breakdown ahead.

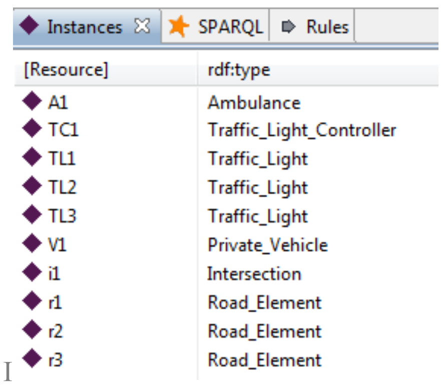
Table 5.6 - Results of SPIN functions and the SPIN template for each time stamp

Time stamp	B1's relative location	V2's relative location	Value of the bVehicleStationar -yExisted variable	Value of the bInTheSituation variable	Working clause of the Update Query Template
Figure 4.9 b	on M25, to Junction 11, 5500 meters	on M25, to Junction 11, 6450 meters	False	True	only INSERT clause
Figure 4.9 c	on M25, to Junction 11, 5500 meters	on M25, to Junction 11, 5650 meters	True	True	None
Figure 4.9 d	on M25, to Junction 11, 5500 meters	on M25, to Junction 11, 5450 meters	True	False	only DELETE clause

5.6 Extensibility of the VEIN application ontology model

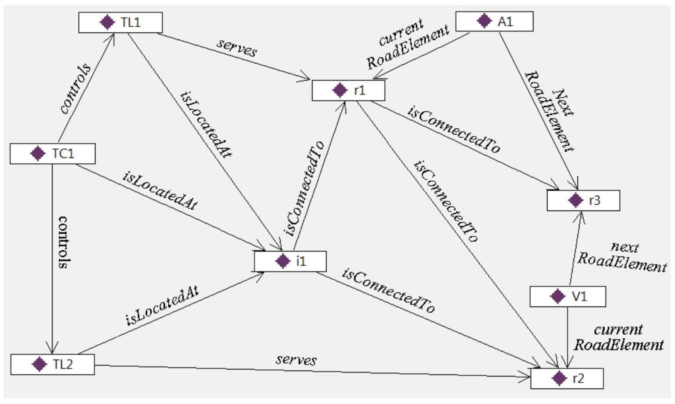
So far, we have followed the development of the VEIN application ontology (i.e. classes, properties, and queries) in order to cover the aforementioned scenarios focusing on vehicle-to-vehicle communications. The application ontology can be easily extended to vehicle-to-infrastructure communications by adding some classes and properties referring to the domain/task ontology. To emphasise the extensibility of the application ontology, this section deals with possible situations with traffic lights, a second emergency vehicle, and multi-lane roads.

First, the VEIN application ontology can be extended to vehicle-to-infrastructure communications by adding some classes, properties, and queries referring to road facilities, such as traffic lights. Classes and properties can be added to describe additional features and their relations in a different situation. Based on triple (subject-predicate-object) patterns of ontology, a situation can be described in a graph-based network model, in which each node has some links, based on triple patterns (subject-predicate-object) of ontology. Nodes represent instances of classes, and links represent their binary relations. The network model describing a situation can be extended by using additional triples. For example, if there are traffic lights and a traffic controller at an intersection (e.g. intersection `i1` is used in the ambulance scenario and is illustrated by Figure 4.5), the situation can be described with instances of additional classes (e.g. `Traffic_Light` and `Traffic_Light_controller`) and their binary relations (e.g. `serves` and `controls`), as shown in Figure 5.19.



[Resource]	rdf:type
A1	Ambulance
TC1	Traffic_Light_Controller
TL1	Traffic_Light
TL2	Traffic_Light
TL3	Traffic_Light
V1	Private_Vehicle
i1	Intersection
r1	Road_Element
r2	Road_Element
r3	Road_Element

(a) Instances



(b) Binary relations among instances

Figure 5.19 - Instances and their binary relations within a scenario

Query logics that extract information related to road facilities (e.g. traffic lights) can also be easily implemented as SPARQL queries. The additional queries, which represent additional logics, will be compatible with the already developed SPIN functions and templates of the VEIN ontology since the logics of the VEIN ontology have been developed from SPARQL queries. The Structured Query Language (SQL) is suitable for querying and extracting data from tabular and structured representations. Meanwhile, SPARQL is a language for querying collections of triples (subject-predicate-object) in a graph-based model, so that it can traverse relationships easily and explicitly in RDF/OWL (Melton, 2006). By using binary relations between traffic lights and a traffic light controller, between traffic lights and road elements, and between vehicles and road elements, context information can be collected from various viewpoints (i.e. the ambulance's point of view and/or the traffic light controller's point of view).

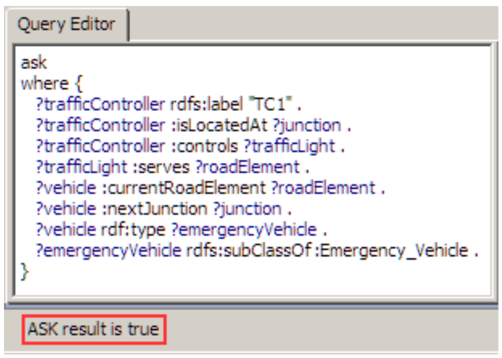
For example the queries in Table 5.7 have been developed to support an effective communication strategy between an ambulance and a traffic controller in an intersection. The first query of Table 5.7 asks whether there are ambulances at the roads towards an intersection where the traffic controller `TC1` is located. The second query of Table 5.7 asks which signal controller the ambulance has to communicate with in order to request a signal pre-emption. In these queries, the ambulance's `nextJunction` property represents the ambulance's direction towards the traffic controller `TC1`. Departing from SELECT statements, an ASK form can be used for a similar purpose, but it just returns true or false. Figure 5.20 shows how binary relations and a class hierarchy can be used to describe possible relations between emergency vehicles and a traffic light controller in an ASK query. If there are any emergency vehicles (i.e. ambulance, fire engine, or police car) on the roads that the traffic light controller covers, the result of the ASK query will be true. It may be also useful to check whether the traffic controller has to take action or not.

Table 5.7 - SPARQL queries to support the ambulance scenario in the third condition

Purpose	SPARQL query	Result

To check whether there are ambulances towards the traffic controller	SELECT ?ambulance
	WHERE {
	  ?trafficController rdfs:label "TC1".
	  ?trafficController :isLocatedAt ?junction.
	  ?trafficController :controls ?trafficLight.
	  ?trafficLight :serves ?roadElement.
	  ?ambulance :currentRoadElement ?roadElement.
	  ?ambulance :nextJunction ?junction.
	  ?ambulance rdf:type :Ambulance.
	}
		A1

To find a traffic controller that the ambulance has to communicate with	SELECT ?trafficController
	WHERE {
	  ?ambulance rdfs:label "A1".
	  ?ambulance :currentRoadElement ?roadElement.
	  ?ambulance :nextJunction ?junction.
	  ?trafficLight :serves ?roadElement.
	  ?trafficController :controls ?trafficLight.
	  ?trafficController :isLocatedAt ?junction.
	}
		TC1

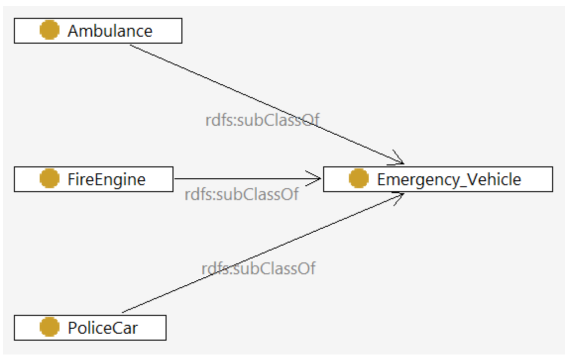


Query Editor

```
ask
where {
  ?trafficController rdfs:label "TC1" .
  ?trafficController :isLocatedAt ?junction .
  ?trafficController :controls ?trafficLight .
  ?trafficLight :serves ?roadElement .
  ?vehicle :currentRoadElement ?roadElement .
  ?vehicle :nextJunction ?junction .
  ?vehicle rdf:type ?emergencyVehicle .
  ?emergencyVehicle rdfs:subClassOf :Emergency_Vehicle .
}
```

ASK result is true

(a) An ASK query



(b) Subclasses of the `Emergency_Vehicle` class

Figure 5.20 - An ASK query (i.e. 'is there an ambulance moving towards the traffic controller?') and the result

Second, the VEIN application ontology can be extended to deal with a second emergency vehicle in a situation. In the ambulance scenario, only one emergency vehicle was considered in the situation, and its request zone for asking its priority to neighbour vehicles was set fixed. However, in the situation of a second emergency vehicle, if the distance between the first emergency vehicle and the second emergency vehicle is close enough, their emergency rule (give away request) cannot be treated separately. The two emergency vehicles may provide one merged emergency rule to the neighbour vehicles to minimise the possible confusion caused by multiple emergency vehicles near each other.

An emergency vehicle sends its ‘give way’ request to the neighbour vehicles only when the vehicles are inside its request zone to minimise its negative impact on the traffic flow. However, emergency vehicles can communicate amongst themselves to share their presences and movements in preparation for a possible collaboration. If an emergency vehicle checks a second emergency vehicle’s presence nearby, it needs to expand its request zone temporarily to consider the second emergency vehicle’s presence. In the VEIN application ontology, it is possible by changing the ‘distance range’ argument of the SPIN update template (e.g. vein:updateInformationFromEmergencyVehicleDSRCMessage) (Figure 5.13).

Table 5.8 provides an example of how an emergency vehicle can check for a second emergency vehicle’s presence, and expand its request zone for asking priority to the neighbour vehicles temporarily.

Table 5.8 - Possible SPARQL queries to support the situation of a second emergency vehicle

Purpose	SPARQL query

To check the presence of a second emergency vehicle and s and their distance	`SELECT Distinct ?ev2 ?ev2dist`
	`WHERE {`
	`    ?emergencyVehicle rdfs:subClassOf :Emergency_Vehicle .`
	`    ?ev1 rdf:type ?emergencyVehicle .`
	`    ?ev1 :myself true .`
	`    ?ev2 rdf:type ?emergencyVehicle .`
	`    FILTER (?ev1 != ?ev2).`
	`    ?ev1 :currentRoadElement ?currentRoadElement .`
	`    ?currentRoadElement :isConnectedTo ?connectedRoadElement .`
	`    {?ev2 :currentRoadElement ?currentRoadElement}`
	`    UNION {?ev2 :currentRoadElement ?connectedRoadElement .`
	`    }.`
	`    BIND(:getDist(?ev1, ?ev2) as ?ev2dist).`
	`}`

To change the ambulance’s request range temporarily considering the second emergency vehicle	`BIND (smf:if(?ev2dist <= ?distanceRangeToAskPriority * 2),`
	`    ?distanceRangeToAskPriority * 2, ?distanceRangeToAskPriority))`
	`as ? distanceRangeToAskPriority).`

Third, even though the VEIN application ontology emphasised vehicles' longitudinal interrelation rather than vehicles' lateral interrelation, it can be extended to deal with a multi-lane situation. The application ontology was developed as a vehicular communication message model for the ambulance scenario and the breakdown scenario on top of a one-dimensional linear road network. An ambulance's goal is to bypass traffic at the highest possible speed to save a life, and it can be done safely and efficiently when the nearby vehicles give the ambulance a clear lane to pass (or give the ambulance more room to travel by stopping or pulling over if the roads are crowded). Accordingly, it is fair to say that the emphasis on the vehicles' longitudinal interrelation was appropriate. Even though two-dimensional or three-dimensional descriptions such as roadway width and multi-lane properties were not considered, the VEIN application ontology model can be extended easily to support vehicles' lateral interrelations. For example, with a new property `vein:lane` and a new variable `?onTheSameLane`, the SPIN functions and the SPIN templates of the VEIN application ontology can be amended (Table 5.9).

In the breakdown scenario, it was assumed that the broken car pulled into the left lane and stopped as far to the left as possible to minimise its adverse effect on the traffic flow. A warning of the broken vehicle was received by drivers in the inside lane approaching the breakdown. They were then required to change lanes as far away as possible from the breakdown if the traffic flow allows. Even though the VEIN application ontology for the breakdown scenario was focused on their longitudinal distance, for a clearer description of the breakdown case, the '`vein:lane`' property can be used.

Table 5.9 - Possible lane property and variable for a multi-lane situation

Description	Source code
Vehicle V1's road information	?myVehicle vein:currentRoadElement ?myCurrentRoad . ?myVehicle vein:lane ?myLane .
Ambulance A1's road information	?ambulVehicle vein:currentRoadElement ?ambulCurrentRoad . ?ambul vein:lane ?ambulLane .
New variable ?onTheSameLane	BIND (smf:if((?myCurrendRoad = ?ambulCurrentRoad) &&(?myLane = ?ambuLLane), true, false)) as ?onTheSameLane).

It has been demonstrated how classes, properties/relations, and query logics in the ontology model can be extended for vehicle-to-infrastructure communications, a second emergency vehicle, and a multi-lane situation. Based on the additional classes and their semantic relations, a situation can be described in more detail, and additional information and knowledge can be extracted from the ontology model by using SPIN functions and templates as well as SPARQL queries.

5.7 Summary and discussion

This chapter outlined the VEIN ontology model from three different perspectives, that is, for the domain ontology, the task ontology, and the application ontology, which were described in Section 5.2, Section 5.3, and Section 5.4, respectively. It also demonstrated how vehicles and their interactions can be instantiated using the VEIN application ontology in Section 5.5.

First, Section 5.2 showed a domain ontology for an ITS environment. The four upper classes of the domain ontology represented the four main classes of the ITS architecture. Even though this domain ontology provides classes only for the general road users, such as pedestrians, cyclists/bicycles, motorcyclists/motorcycles, and vehicles, it can be extended to other road users (e.g. powered wheelchairs, mobility scooters, horse riders) for a specific purpose. The domain ontology contains a class for level crossings (railroad crossing) where a road intersects a railway line without recourse to a bridge, but it does not consider transferring between transport modes. To use this domain ontology for intermodal passenger transport, which involves other transport modes (e.g. airways, railways, waterways), transferring facilities, such as train stations, airports, and piers, can be added.

Second, in Section 5.3, the task ontology was developed to model communications among the four upper classes (i.e. vehicles, infrastructure, travellers, and centres) of the domain ontology. The `CommunicationAgent` class and the `CommunicationMessage` class are extracted from the data flows representing the actual information exchanged across subsystems of the `Vehicle` class and the `Infrastructure` class. Then, classes for communication event and communication network are added to represent the communication itself and communication connection, respectively. The task ontology can be seen as a conceptual bridge between the domain ontology and the application ontology representing vehicular communications.

Third, Section 5.4 showed the application ontology of the VEIN model to support the situations of the two scenarios. It focused on properties and relations rather than classes and their hierarchy to describe a vehicle’s route and relative location. It also provides SPIN functions and templates to update a vehicle’s route and location and keep them as asserted information. When a situation is described with the instances of classes and their properties, a set of triples (declarative sentences) may represent the situation. Similarly, a vehicle's states in the situation can be seen with the directly related triples. In addition, each triple can be used for further knowledge and reasoning for the vehicle to take action, especially in partially observable environments.

Fourth, Section 5.5 demonstrated how the interactions between vehicles for each snapshot of the scenarios can be implemented by DSRC messages and the VEIN application ontology. Each vehicle has its own ontology model, which inherits classes, properties, SPIN functions, and SPIN templates from the VEIN ontology model. Each vehicle itself is instantiated in its own ontology model while it can instantiate other vehicles in its model when it receives a DSRC message from them. It also shows that the ontology model can be updated if DSRC messages deliver arguments of a query template. Additionally, it may deliver a whole query or partial ontology to update the ontology model depending on the applications.

Fifth, Section 5.6 highlighted the extensibility of the ontology model for the OBE-to-RSE communications and the situations of a second emergency vehicle and a multi-lane road. The graph-based representation made the ontology model flexible and extensible so that instances of road facilities and their properties/relations are easily added, and new information can be extracted by additional queries. Apart from that, new properties for road elements can be added. Currently, there is no property for the road element's directional information in the current ontology model, while a vehicle's direction on a road element can be acquired with its `previousJunction` and `nextJunction` properties. Each vehicle's ontology model was intended to keep minimum ground facts for the vehicle's current state. However, road elements' directional information is compulsory to assign a vehicle's route. The ontology model can be extended to support a situation that a vehicle changes its route to avoid a traffic congestion (Desai et al., 2013) by defining additional properties (e.g. directional information, restriction information) to the road element class. Properties, which are not even known at the design time, can be easily added, and this flexible extensibility is a benefit of using an ontology model.

In summary, this chapter proposed the VEIN ontology model that consists of classes, instances, and object properties to describe vehicles' movements in the road environment. For the VEIN application ontology, SPIN functions and templates are built in order to implement the information update from the communications among vehicles via OBE. Consequently, the VEIN model provides a single ontological framework from the domain ontology and task ontology supporting general ITS environments to the application ontology to cover the ambulance scenario and the vehicle breakdown scenario. The VEIN ontology model also uses a relative description for the vehicle's locations in order to simplify the decision-making logic (to check whether a vehicle needs to take action in an emergency situation or not) by splitting the geometric calculation from the decision-making process. Even though the VEIN ontology model has focused on the interaction between the vehicles of the two scenarios, it covers both a declarative approach that describes vehicles' movements and a procedural approach that describes the rules of a system or desired behaviours of a vehicle in emergency situations. Therefore, it can be easily extended for more complex interactions among vehicles and the infrastructure.

This chapter highlighted sharing vehicles' presence (i.e. partial route and relative location) by sending/receiving their partial ontology as the contents of DSRC message. However, it did not deal with the volume issues of the ontology model and the DSRC message set. The large volumes of the ontology model and the DSRC message set may delay vehicles' processing and communication respectively, and they can have a negative effect on vehicles' situation awareness. Thus, it is better for each vehicle to maintain minimum volumes of the ontology model and the DSRC message set. To keep minimum ground facts in each vehicle's ontology model to maintain the vehicle's current state, the ontology model can keep minimal road information. The road information consists of only its own route, connected roads, and their link–link topology, in memory in real-time. To minimise the volume of DSRC message set, vehicles can share their partial route by using unique identifiers (i.e. URIs) of road elements and junctions. There is no need for all vehicles to use the same map database in order to use road features' identifiers, but it is necessary that each road/junction element on the map databases can be referred with its own identifier. Under the assumption that vehicles can refer same road features with same identifiers, vehicles can share their relative location throughout vehicular communications properly.

6. Agent modelling and simulation

6.1 Introduction

The previous chapter proposed the VEIN ontology model as a messaging model emphasising the information exchanges by DSRC, and the static snapshots of the scenarios were used to validate the messaging model. This chapter outlines vehicles' movement and communication events in a dynamic virtual road environment that is modelled by Agent-Based Modelling and Simulation (ABMS). This chapter presents the third stage of the methodological framework (Figure 1.3), that is, modelling and simulating intelligent vehicles and their communications in a virtual road environment.

The rest of this chapter comprises six sections. Section 6.2 provides information that is essential for modelling vehicle agents' characteristics and their movements in the road environment. Section 6.3 describes the modelling process for the virtual road environment, in which vehicle agents move and interact with each other by sharing communication messages. Focusing on two scenarios, this section outlines the purpose of the simulation, the virtual road environment as a modelling space, the model's assumptions, and the vehicle agent's attributes and behaviours. Following this, Section 6.4 considers the development of the simulation environment using Repast Simphony from its context, projection, and class hierarchy to setup the procedure, dynamic procedure, and the output data of the simulations. Section 6.4 also includes a description of the logic for the vehicles' communications based on their locations and movements following predefined routes on the road network. Section 6.5 describes the preliminary simulations in order to verify the algorithms for vehicles' movements and communications and to decide the number of simulation runs. Section 6.6 presents the simulation results and statistics using different initial parameters (e.g. different portions of OBE-implanted vehicles in situations). The simulation results are analysed to identify the effect of vehicular communications on the efficiency pattern of traffic in different simulation settings (i.e. different portions of OBE-implanted vehicles and non-OBE vehicles) in the two scenarios. Finally, Section 6.7 summarises the chapter and concludes with an overview of the main issues discussed in this chapter.

6.2 Vehicle agents and the road environment

There are two kinds of emergency events in terms of vehicular movement in the road environment. The first refers to ‘static events’, which have a fixed location or coverage (e.g. a car breakdown, road work ahead, or foggy area), and the second refers to ‘dynamic and spatially moving events’, such as an emergency vehicle that requires the area to be clear around it. When a vehicle is heading towards a static event, or when a dynamic event is coming towards a vehicle, the vehicle should take action to resolve the situation. If beforehand, the vehicle receives, through vehicular communication, information with respect to an event, then it will have more time to respond to this specific event.

These two different kinds of emergency events were described as two scenarios in Section 4.3. In the scenarios, it is assumed that vehicles can communicate with each other to share information about their presence and traffic events. The agent simulation is designed to provide a virtual environment of these emergency events, in which vehicle agents can reduce their speeds safely to be ready in advance, having been alerted via vehicular communication, and so avoid conflict with the events or pass through the events safely.

This section discusses the modelling process of intelligent vehicles and their environment. Specifically, it explains intelligent vehicles’ characteristics including their movements and communications based on their relative location and route. Then, the road environment, where vehicles are moving, is outlined, focusing on how it can be created, using geographic information such as road network and facilities locations (e.g. hospitals for the ambulance scenario), for the simulation purposes.

6.2.1 Vehicles as intelligent agents

Vehicles move and interact with each other as agents in the road environment, while during the simulation, the individual behaviours of each vehicle can be measured. A benefit of agent-based modelling is that a whole traffic system can be explained and analysed based on vehicles' individual behaviours (i.e. movements and interactions). Using vehicular communications, vehicles are informed beforehand about a situation, so they have a longer response time to prepare and take action to deal with the situation. The effects of these communications amongst vehicles in an emergency situation can be measured only at the individual level because in this research, short-range vehicular communication (i.e. DSRC) has been used for a vehicle to share information about its presence and situation with neighbouring vehicles. In addition, the coalition of vehicles for the communication constantly changes as vehicles move while following their routes, and vehicles in the vicinity of the emergency situation participate in the communication only to resolve the situation.

The focus here is on a vehicle agent's internal states and several road traffic statistics. The vehicle agent's internal states represent the driver's mental states, which are related to his/her next action/behaviour. The road traffic statistics includes national speed limits, recommended minimum stopping distance, and free-flow[24] speeds in order to model vehicles' speed and inter-vehicle distance.

6.2.1.1 Vehicle's internal states

As mentioned in Section 3.2.1, an agent's characteristics can be described using a BDI architecture, and an agent's beliefs, desires, and intentions can be mapped onto its knowledge-base, goals, and plans for action, respectively. Building a vehicle agent's BDI model that represents the driver's knowledge, goals, and intentions in a general situation would be a complicated and challenging process, and indeed, building such a model is beyond the scope of this research. This research deals only with vehicular interactions by communication in emergency situations, and in an emergency situation, most vehicles will take action in order to resolve the situation by following the emergency rules. Therefore, resolving the emergency situation is regarded as the most important goal temporarily, so a driver's BDI model can be simplified.

Since the simulation focuses on the vehicles' communications and movements during the emergency situation, the vehicle agent's behaviours before the emergency situation are not measured. Instead, throughout the simulation and before the emergency situation, it is assumed that the different speed and location of each vehicle are represented by the driver's behaviours.

Generally, an agent will act based on the intention to achieve a goal from its priority list of goals, while these goals and intentions change over time according to the agent's internal states and

[24] Free-flow speed is the term used to describe the average speed that a driver would travel if there were no congestion or other adverse conditions (e.g. bad weather).

external variables (i.e. the environment's external variables and neighbour agents' internal states). So, an agent will define a goal and form a plan for achieving this goal based on its current internal states and the external variables. However, when a vehicle agent encounters an emergency event, this becomes the most important short-term goal for the vehicle to achieve irrespective of its individual objectives, and it will take action to resolve the emergency situation. In this context, an agent can sense an emergency situation, and its intentions can be reactions, such as stop, slow down, move again, or change lanes. Therefore, when considering emergency situations, an agent's BDI architecture could be simplified by using the agent's memory, policy, and performance (Table 6.1).

Table 6.1 - A vehicle agent's BDI in an emergency situation

BDI	General content	Possible content in an emergency	Simplified term
Beliefs	Internal knowledge of the world	The vehicle's and neighbour vehicles' location/ speed/ route on the road network in an emergency situation	**Memory** to understand current traffic situation
Desires	Goals to achieve	Generally accepted reactions to resolve the emergency situation without any conflict over the event	**Policy** to act without the conflict over the emergency rule
Intentions	Most important goals, most doable plans	Resolving the emergency situation = the most important goal (temporarily)	**Performance (behaviour)** to resolve the situation

In agent-based modelling, perception, performance (behaviour), memory, and policy are the agents' four important characteristics (Abdou et al., 2012), so vehicles that communicate with each other in an emergency situation can be considered and modelled as agents. Even though vehicles are following static emergency rules in an emergency situation, each vehicle's behaviours will differ depending on its internal states, such as its relative location, speed, and route.

As was noted earlier, vehicle agents' behaviours are limited and simplified only for actions to resolve an emergency situation, which is the most important temporary goal for vehicle agents to achieve. So, the vehicle agents are designed to measure the effect of vehicular communications in an emergency situation by sharing the information related to their presence and current traffic events. Even though all general aspects of drivers' behaviours are not considered, the limited behaviours of vehicle agents are enough to build a simulation environment for different proportions of communicative[25] vehicles in an emergency situation.

6.2.1.2 Vehicle's speed and inter-vehicle distance

To simulate proper vehicle density on the road, it is crucial to set a specific number of vehicles because the coverage of the road network will be fixed. Since vehicles are moving at high speed on the road network, their speed and inter-vehicle distance are also important variables. As road environments have been designed for road users, mainly cars, there are traffic instructions and recommendations for a vehicle's speed and braking distance based on roads' type, geometry, circumstance, etc. The speed of vehicles and the recommended stopping distances can be referred to in order to simulate an ideal road environment that has a proper vehicle population and that is close to the road's designed capacity.

On the one hand, speed limits play 'a fundamental role' as 'an indicator of the nature and risks posed by that road' to motorised and non-motorised road users (Department for Transport, 2006, p.2). If there are no congestion or other adverse conditions, the average speed of vehicles on roads is closely related to the speed limits of the roads (Table 6.2). In the simulation, measured free-flow car speed ranges in Table 6.2a and Table 6.2d are used to randomise vehicles' initial speeds for the simulation of the ambulance scenario and the breakdown scenario, respectively. Even though vehicles' speed in the simulation can be set at greater than the national speed limits, it is important to note that the speed limit is not a target speed for all circumstances. In reality, for safety reasons, drivers need to slow down when they approach a bend in the road, a roundabout/junction, roadworks, and so on. They also need to reduce their speed not only when they share the road with other road users (e.g. pedestrians, motorcyclists, etc.), but also when they drive at night or in bad weather.

On the other hand, each vehicle needs to leave enough space from the vehicle in front, in order to be prepared for a sudden change in the front vehicle's behaviours (e.g. when the front vehicle slows down or stops). There are recommended stopping distances based on vehicles' speed, and even in faster-moving traffic flows. A vehicle needs to allow at least a two-second gap from the

[25] A communicative vehicle is a car that has a vehicular communication device, so that it can communicate with other communicative vehicles and road facilities to send and receive information about a traffic situation.

front vehicle for its own safety (Driving Standards Agency, 2007). If the vehicle travels at the speed of 30 mph, its ideal stopping distance from the front vehicle is 23 metres (Figure 6.1). On a one-way single-track road with a 2300 metres length and where 100 vehicles with the speed of 30 mph are randomly located, it can be said that their average distance may be similar to the typical stopping distance of 23 metres. Meanwhile, the road network of a local area in reality is composed of a number of linear features, and the road environment can be treated as a two-dimensional space, so the average distance between vehicles should be different from the above case of the long single-track road. Therefore, after vehicles are simulated, the average inter-vehicle distance needs to be checked, and then the number of vehicles can be adjusted to get a rational average inter-vehicle distance by a process of trial and error.

Table 6.2 – Free-flow car speed ranges in 2010 (Department for Transport, 2011a; Department for Transport, 2011c)

Vehicle speed of cars	Percentage
Under 20 mph	5 %
20-29 mph	*49 %*
30-34 mph	*30 %*
35-39 mph	*12 %*
40-44 mph	*3 %*
45-49 mph	*1 %*
50 mph and over	*0 %*
More than 5 mph over limit	*16 %*
Average speed (mph)	30 mph
Number observed (thousands)	54,544

(a) on Built-up roads (30 mph limit)

Vehicle speed of cars	Percentage
Under 20 mph	1 %
20-29 mph	3 %
30-39 mph	17 %
40-49 mph	44 %
50-59 mph	28 %
60-64 mph	5 %
65-69 mph	2 %
70 mph and over	*1 %*
More than 10 mph over limit	*1 %*
Average speed (mph)	47 mph
Number observed (thousands)	42,251

(b) on Single carriageways (60 mph limit)

Vehicle speed of cars	Percentage
Under 30 mph	0 %
30-39 mph	*0 %*
40-49 mph	*3 %*
50-59 mph	*17 %*
60-64 mph	*17 %*
65-69 mph	*21 %*
70-79 mph	*32 %*
80 mph and over	*10 %*
More than 10 mph over limit	*10 %*
Average speed (mph)	68 mph
Number observed (thousands)	43,847

(c) on Dual carriageways (70 mph limit)

Vehicle speed of cars	Percentage
Under 50 mph	4 %
50-59 mph	*14 %*
60-64 mph	*14 %*
65-69 mph	*19 %*
70-74 mph	*21 %*
75-79 mph	*15 %*
80-89 mph	*12 %*
90 mph and over	*2 %*
More than 10 mph over limit	*14 %*
Average speed (mph)	69 mph
Number observed (thousands)	385,917

(d) on Motorways (70 mph limit)

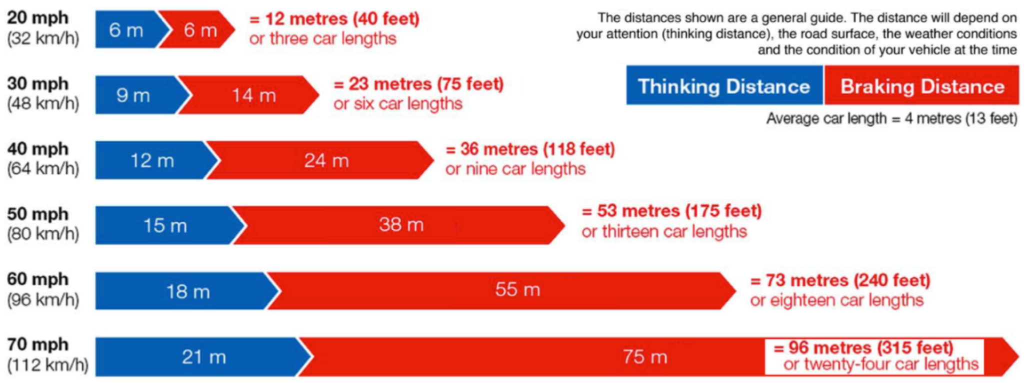


Figure 6.1 - Recommended minimum safe stopping distance (Driving Standards Agency, 2007)

In our daily lives, the majority of vehicles exceed the speed limits and occasionally do not maintain the necessary stopping distances (Brake, 2010; Department for Transport, 2011b; Department for Transport, 2011d). In the simulation, an ideal road environment, in which vehicles travel at a desired speed if they can and maintain a safe stopping distance from the front vehicle, is considered, and the free-flow vehicle population for each simulation is set mathematically. A vehicle travels at a desired speed if there is no vehicle in front, but the vehicle needs to slow down to keep a safe distance from the lead vehicle. There are several car-following models (e.g. Krauss Model, Nagel and Schreckenberg Model, Wiedeman Psycho-Physical Model, General Motors Model, Gipps Model, Intelligent Driver Model) to define acceleration/deceleration function of a vehicle's dynamic behaviour when approaching the lead vehicle (Härri et al., 2007). In this research, the time-continuous Intelligent Driver Model (IDM) is adapted to produce realistic acceleration profiles and reactions of vehicles in a single lane traffic situation, and its mathematical description is presented in Table 6.3 (Treiber and Kesting, 2013).

To model vehicles approaching a stationary vehicle (for the broken car in the second scenario) or a junction (for the first scenario), a traffic light model or an junction/intersection management model is necessary. A simple non-signalised junction management model is implemented based on the dynamic term $v\Delta v/(2\sqrt{ab})$ of the IDM and the first-in-first-out (FIFO) principle. The dynamic term can be simplified and seen as the kinematic deceleration of a vehicle when approaching to a stationary vehicle or a junction.

Table 6.3 – Mathematical description of IDM

Equation	$\dot{v} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v,\Delta v)}{s}\right)^2\right]$ $s^*(v,\Delta v) = s_0 + max\left(0, vT + \frac{v\Delta v}{2\sqrt{ab}}\right)$ s: the current distance, s^*: the desired distance	
Parameters	Road	Motorway
Desired Speed v_0 at free flow	*See* Table 6.2*a*	*See* Table 6.2*d*
(When followed by other vehicles)	*30 mph*	*60/70/80 mph*
Time gap *T*	*1.0 s*	
Minimum gap S_0	*2 m*	
Acceleration exponent δ	*4*	
Acceleration *a*	*1.0 m/s²*	
Comfortable deceleration *b*	1.5 *m/s²*	
Vehicle length	*5 m*	

Meanwhile, in a general situation, vehicles travel according to their own goals and plans as long as their movements are not strongly against traffic instructions. It is possible to model individual-level behaviours according to statistics and surveys about drivers' behaviours and attitudes. However, this is still one of the most difficult aspects of agent-based modelling and simulation, and as stated earlier, it is beyond the scope of this research, since it is a separate research question that deals with general traffic analysis and requires the implementation of a different simulation approach.

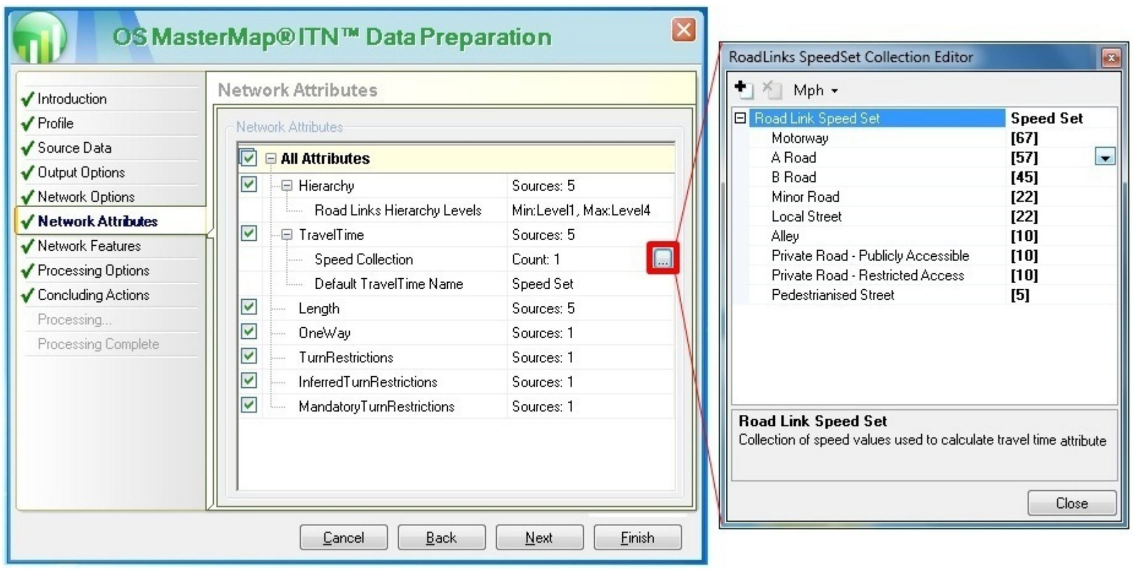
Our focus here is on vehicles' actions through vehicular communications in emergency situations, in which emergency rules can be generally accepted to resolve the situations. Therefore, the simulation itself is intended to provide a virtual road environment, in which vehicles send/receive traffic information and have more time to prepare for an emergency situation. To measure the effect of vehicular communications in an emergency situation, the simulation has to be executed based on the assumption that there is proper number of vehicles and no other traffic obstacles, such as a traffic jam, except the emergency event itself. In addition, the simulations focus on communications from an emergency vehicle or a broken-down vehicle to the vehicles in the vicinity, rather than communications amongst neighbouring vehicles.

Here we have presented some statistics and measurements that can be used to set the vehicles' properties (e.g. speed, inter-vehicle distance) in the simulation. The next section outlines the road environment and its initial dataset in more detail.

6.2.2 The road environment

As a model space, the geometric information of a virtual road environment is necessary for locating the vehicle agents on it and simulating their movements and interactions. The geometry of a road network and routing information within the M25 area were acquired from the OS MasterMap Integrated Transport Network (ITN) layer. ArcGIS 10 and Productivity Suite 2.1 were used to get a feature dataset and a network dataset from the OS MasterMap ITN data. The ITN layer contains Great Britain's road network, and there are three main elements: Road Network, Road Routing Information (RRI), and Urban Paths (Ordnance Survey, 2011).

First, the Road Network theme represents the geometry of Great Britain's road network from motorways, A-roads, and B-roads to local streets, pedestrianised streets, and alleys. Road network analysis supports various kinds of network analysis, for example, routing for a shortest path, servicing a set of orders, finding a closest facility, identifying a service area, and finding the allocation of facilities. For the purposes of the previously noted analyses, there are two ways of calculating the vehicles' accessibility: a) using travel distance and b) using travel time. To use travel time for the network analysis, it is necessary to use general speeds for the roads when building the road network, as shown in Figure 6.2.



Network Attributes

All Attributes	
Hierarchy	Sources: 5
Road Links Hierarchy Levels	Min:Level1, Max:Level4
TravelTime	Sources: 5
Speed Collection	Count: 1
Default TravelTime Name	Speed Set
Length	Sources: 5
OneWay	Sources: 1
TurnRestrictions	Sources: 1
InferredTurnRestrictions	Sources: 1
MandatoryTurnRestrictions	Sources: 1

Road Link Speed Set	Speed Set
Motorway	[67]
A Road	[57]
B Road	[45]
Minor Road	[22]
Local Street	[22]
Alley	[10]
Private Road - Publicly Accessible	[10]
Private Road - Restricted Access	[10]
Pedestrianised Street	[5]

Road Link Speed Set
Collection of speed values used to calculate travel time attribute

Figure 6.2 - The default speed data for roads to calculate the travel time attribute in the road network

Second, RRI includes routing information for drivers on mandatory and banned turns and other restrictions. The routing information of RRI is also categorised into three qualifiers: environmental qualifiers, vehicular qualifiers, and date/time qualifiers. Environmental qualifiers contain information with respect to one-way streets, level crossings (railroad crossing), bridge heights, traffic calming measures, etc. Vehicular qualifiers describe the types of vehicles affected by the routing information, such as a section of road (e.g. bus use only). Date/time qualifiers show periodic information like seasonal road closures or bus lanes operating at certain times of the day only.

Some environmental qualifiers of routing information are also used to obtain information about one-way and no-entry restrictions. This information is acquired by joining the road links feature class (`RDLK`) and the two tables (`RRIDL`, `ENVQ`) described in Table 6.4. When the feature class and two tables are joined, the `Qualifier` field and the `DirectedLinkOrientation` field illustrate the restriction attribute and its direction, respectively (Table 6.5). The '+' value of the `OneWay` field shows an one-way flow from the beginning to the end of the road link while the '−' value of the field indicates an one-way flow in the opposite way. These Join operations consider only road links where the whole road link has the restriction, so partial road link restrictions such as bus lanes in conjunction with date/time qualifier data are not considered in the simulation.

Table 6.4 - Key fields for join operations to get one way and no entry restrictions (ESRI UK, 2006)

Source Table / FC	Join Field	Target Table / FC	Join Field
RRIDL	TOID	ENVQ	TOID
RDLK	TOID	RRIDL_ENVQ	RRIDL.DIRECTEDLINKTOID

Table 6.5 - Field names to get one way and no entry restrictions (ESRI UK, 2006)

Attribute	Source Fieldnames	Description
OneWay	RRIDL.DirectedLinkOrientation where ENVQ.Qualifier = 'One Way'	One way and direction "+" : Permitted flow is in the direction of the digitisation of the line "-" : Permitted flow is against the direction of digitisation of the line "N" : There is no one-way restriction, flow can occur in either direction
NoEntry	RRIDL.DirectedLinkOrientation where ENVQ.Qualifier = 'No Entry'	Indicates that the link is no entry "+" : No entry at the start of the line as digitised "-" : No entry at the end of the line as digitised "N" : The link does not have a no-entry restriction

Third, Urban Paths is a new theme for man-made footpaths, subways, steps, footbridges and cycle paths in urban areas. It was made to join up path and road centrelines by adding the links and nodes of urban paths to the ITN network in order to encourage multi-modal travel in towns and built-up areas. However, as we focus on the movement of vehicles, the Urban Paths theme is not relevant to the proposed simulation.

Even though vehicles can move on most types of roads in the road network - except pedestrianised streets and alleys - in this simulation, three types of major (motorway, A-road, B-road) and minor roads are used as an environment in which vehicle agents can travel (Table 6.6). A-roads, B-roads, and minor roads are used for the ambulance scenario, while motorways and some dual carriageway A-roads are utilised for the breakdown scenario.

Table 6.6 - Road types used in the simulation

ITN road type (DescTerm)	Vehicle access	Used in the simulation
Motorway	✓	✓
A-road	✓	✓
B-road	✓	✓
Minor road	✓	✓
Local Street	✓	
Private Road - Publicly Accessible	✓	
Private Road - Restricted Access	✓	
Pedestrianised Street		
Alley		

Section 6.2 presented the traffic statistics that can be used to model vehicles' properties (e.g. speed and inter-vehicle distance). Even though roads are designed for vehicles to keep within speed limits and to maintain the recommended minimum safe stopping distance, it is assumed that using traffic statistics of vehicle speed and inter-vehicle distance is more realistic than using traffic regulations and recommendations for modelling moving vehicles on the road. We have also outlined the process of building a virtual road environment and explained how a directed road network was extracted and simplified for vehicle agents.

6.3 Agent modelling

This section attempts to bridge the communication massage model that was developed from the scenarios and the agent simulation by presenting the dataset used to build the virtual road environment as well as the vehicle agents' attributes and behaviours. It begins by clarifying the purpose of the agent simulation, which is to examine the model for vehicular communications in dynamic situations (Section 6.3.1). Then, Section 6.3.2 describes the virtual road environment for the ambulance scenario and the vehicle breakdown scenario. Section 6.3.3 presents the modelling assumptions, while Section 6.3.4 outlines the vehicle agents' attributes and behaviours.

6.3.1 The purpose of the simulation and expected results

This simulation is designed to provide a dynamic road environment where an ambulance or a broken vehicle communicate with its neighbour vehicles to share its presence and elicit their reactions/behaviours to resolve an emergency situation (i.e. the ambulance scenario and the breakdown scenario in Chapter 4). The concepts of the VEIN application model are reused to develop the vehicle agents' communication behaviour, but the simulation itself does not provide a dynamic validation of the VEIN application ontology. As a dynamic validation of the vehicular communications, the simulation just provides a virtual environment of the dynamic situations that may examine vehicles' possible proactive/reactive behaviour based on the vehicular communications.

Vehicles' communications and interactions can be simulated repeatedly with different parameters, and the results can be used to show the effect of vehicular communications in different situations. We assume that hybrid environments, in which different portions of OBE-implanted vehicles and non-OBE vehicles, are simulated to assess the effect of vehicular communications on the traffic efficiency in emergency situations and to support the transition to the ITS environment for the OBE-implanted vehicles. To achieve this, five types of hybrid environments are simulated for situations of the two scenarios, and each environment represents from 0% of OBE-implanted vehicles to 100 % of OBE-implanted vehicles (Figure 6.3).

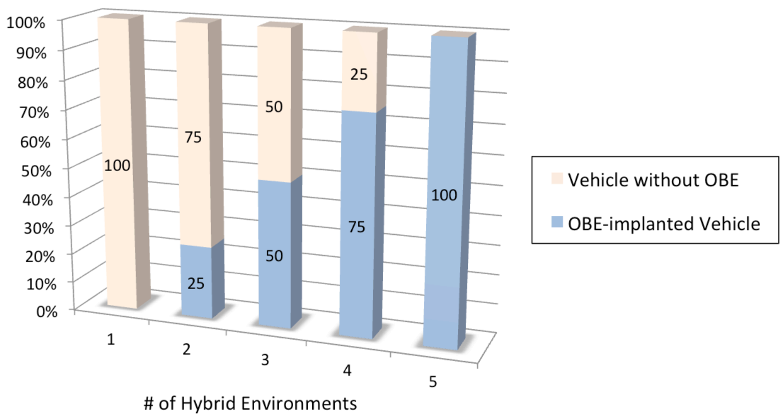


Figure 6.3 - Hybrid environments depending on the percentage of the OBE-implanted vehicles

One benefit of agent modelling and simulation is that it shows macro-level regularities from micro-level individual interactions (Abdou et al., 2012). During the simulations of the scenarios, the total number of affected vehicles will be logged to represent the relationships between the vehicular communications and the effects of the emergency situation to the traffic flow. In this context, the vehicular communications represent micro-level individual interactions, while the effects of the vehicular communications on the traffic flow represent the macro-level regularity.

Therefore, the agent simulations in the five hybrid environments that represent the transition of the percentage from 0% of OBE-implanted vehicles to 100 % of OBE-implanted vehicles are useful to check if there are any positive effects of the vehicular communications to the whole traffic flow. In addition, as mentioned in the introduction, altering existing vehicles and infrastructure to provide such environments in a physical way is prohibitively expensive and difficult to achieve, so futuristic road situations are better to be simulated before any physical implementation actually takes place.

6.3.2 The virtual road environment as modelling space

The virtual road environment has to be sufficiently large to permit enough heterogeneity and opportunities for vehicular interactions, and at the same time, it has to describe the road environment for the situations of two scenarios. The scenarios represent sudden traffic obstacles on the road; the former represents a spatially dynamic obstacle with the example of the sudden appearance of a rapidly moving vehicle from behind or from other directions, and the latter depicts a spatially static but temporally sudden obstacle, such as a sudden vehicle breakdown in front.

Even though there are other kinds of traffic obstacles (spatiotemporally static obstacles, like roadworks), this research focuses on spatially dynamic obstacles and spatially static obstacles on the road, as described in the scenarios (Section 4.3). In the simulation, as the vehicles move and communicate with each other on the road, the virtual road environment is necessary to simulate the vehicles' movements and communications. The following two sections outline the virtual road environments for modelling the ambulance scenario and the vehicle breakdown scenario.

6.3.2.1 The road environment for the ambulance scenario

For the ambulance scenario, the geometric part for the road environment consists of the road network and hospital points. The road network of Greater London was generated from Ordnance Survey MasterMap ITN layer, and using data from the NHS website a point layer was created to represent hospitals in London that have an accident and emergency (A&E) department (Figure 6.4). Three hospitals were chosen from the hospital layer to extract local road networks for each hospital in order to simplify and localise the road environment. Each road network for a hospital represents a different road geometry in urban and suburban areas (Figure 6.5).

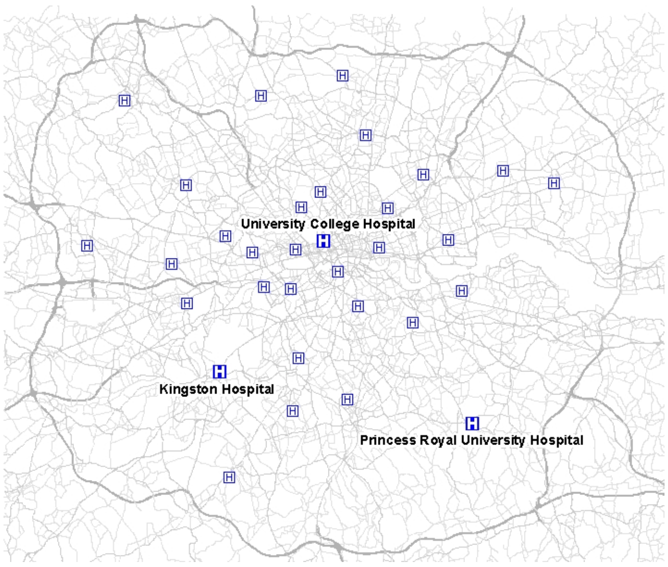
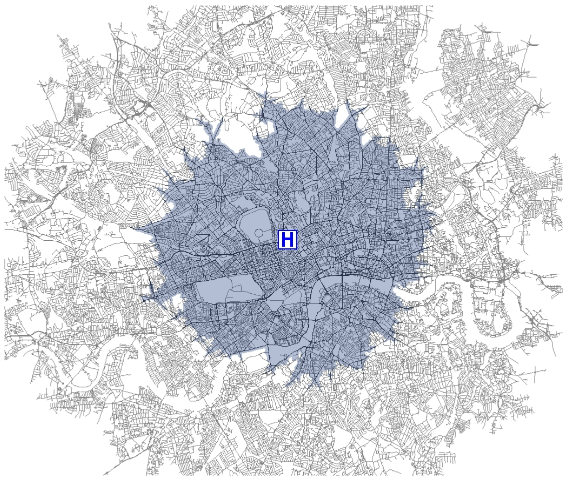
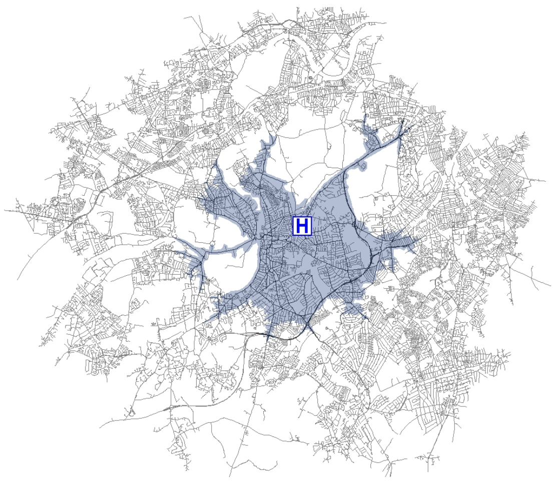
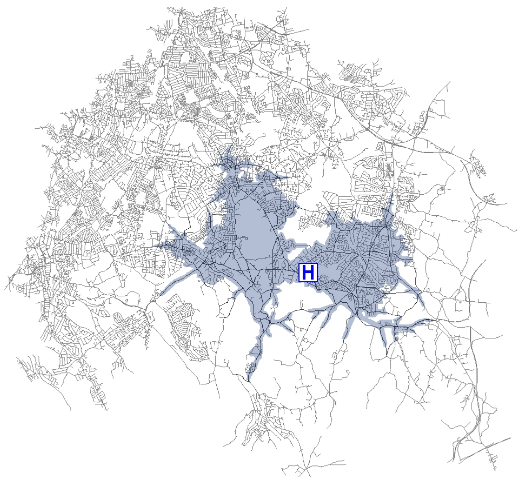

University College Hospital
Kingston Hospital
Princess Royal University Hospital

Figure 6.4 - Hospital locations and the road network within the M25



a) University College Hospital





(b) Kingston Hospital (c) Princess Royal University Hospital

Figure 6.5 - Road networks around three hospitals

According to the national standards for red calls, an emergency response should reach 75 percent of calls within eight minutes for an immediately life threatening condition, such as cardiac arrest (Table 6.7). However, ideally, help by trained ambulance personnel should be provided within five minutes of a cardiac arrest (Pell et al., 2001; Torp-Pedersen et al., 1989). Consequently, service area analysis is carried out to extract local road networks that are accessible to each hospital within five minutes (the shaded areas[26] of Figure 6.5).

[26] Only shaded areas of the road networks are used for the simulations. To create a simpler simulation environment as designed in Table 6.6, local streets are also removed from the road networks.

Table 6.7 - National standards for red calls (NHS Information Centre, 2008)

Red call	Category	Condition	National target for emergency response
Red 8	Category A	Immediately life threatening condition (e.g. cardiac arrest, respiratory arrest)	75% in 8 minutes
Red 19	Category B	Serious but not immediately life threatening condition (e.g. traumatic injuries or fractures)	95% in 19 minutes

6.3.2.2 The road environment for the vehicle breakdown scenario

For the vehicle breakdown scenario on the motorway, the M25 motorway carriageways are extracted from the Ordnance Survey's MasterMap ITN layer. Some dual carriageway A-roads that are directly connected to M25 and/or around motorway junctions are also considered (Figure 6.6). As dual carriageways are used for the road environment of the breakdown scenario, vehicles on one carriageway have no influence on the traffic flow on the other carriageway.

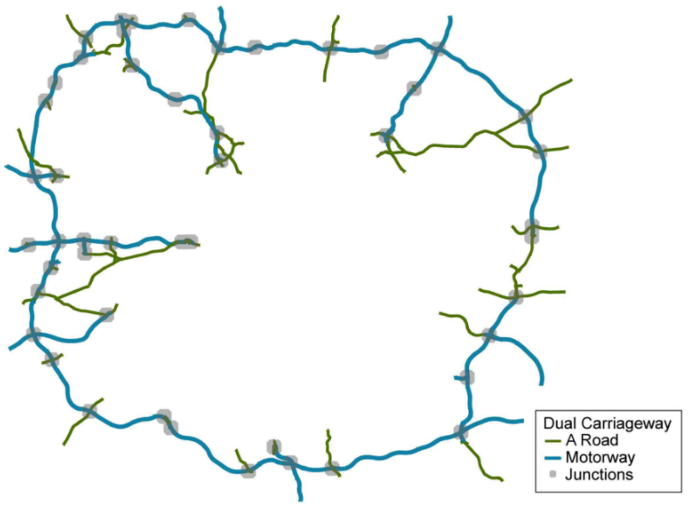


Figure 6.6 - Dual carriageways around London

When importing OS MasterMap ITN data into a feature dataset and a network dataset in a geodatabase, ArcCatalog and Productivity Suite were used. A network dataset in a geodatabase is a virtual feature dataset, which only exists logically for the network analysis in ArcGIS. However, for the simulation in the Repast Simphony, a road network has to be loaded within a shape file format. In the original shapefile, road features around interchanging motorways are split into smaller pieces and a road feature does not represent a motorway segment/junction properly. Thus, road features are edited to express the road connectivity of motorway segments and junctions around two interchanging motorways, as shown in Figure 6.7.

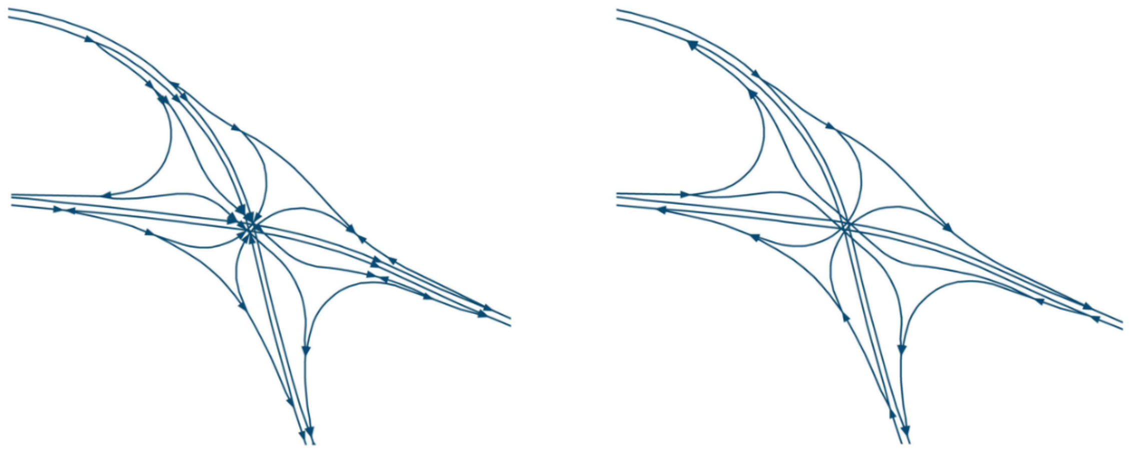

(a) Before editing (b) After editing

Figure 6.7 - An example of road feature editing for the road direction and connectivity

6.3.3 Model assumptions

Generally, when modelling emergent[27] phenomena, existing theories and information are required to support and justify the model, but the simulation in this research is not a case of emergent phenomena. The simulation was designed to supplement the static validation of the communication messaging model, which described the vehicles' movements and communications with static snapshots in the previous chapter. Through the simulation, a dynamic virtual road environment is built to examine the effect of the vehicular communication to support vehicles' interactions by sharing their locations and movements in advance.

It is assumed that vehicles are the only agents for the simulations. In order to simplify the simulations, road facilities, such as traffic signals, are not considered as agents. In addition, to compare the effect of the OBE-implanted vehicles in emergency situations, the portion of OBE-implanted vehicles is treated as the only different parameter for simulations. Other parameters, such as the number of vehicles, and each vehicle's preferred speed, origin, destination, and route, are fixed and remain static.

[27] Emergent is the adjective form of emergence. Emergence is the process of complex pattern formation from a multiplicity of relatively simple interactions. Emergence is central to the theories of complex systems.

Of course, an emergency situation on the road is still in the boundary of the complex transport systems and patterns, which arises from the multiplicity of relatively simple interactions among vehicles and the road infrastructure, so it could be considered as an emergent phenomenon. However, in the simulations many variables are simplified and fixed to examine the potential of the vehicular communication. Thus, this research demonstrates that local interaction amongst vehicles can be supported by information and communication technologies, especially in emergency situations.

6.3.4 Vehicle agents' attributes and behaviours

Vehicle agents share their relative locations via communications, so their attributes and behaviours are related to spatiotemporal information and vehicular communications. Thus, vehicle agents' attributes and behaviours are designed based on the classes and properties of the VEIN application ontology (Table 6.8). Every vehicle agent has general attributes (e.g. current coordinates, current speed, current road element, next junction element, etc.) for travelling. The emergency vehicle and broken-down vehicle have proactive methods for sending messages, while the others have reactive methods to respond by reducing speed, giving way to the ambulance or changing lanes to avoid the broken-down vehicle.

For vehicles that are travelling, each vehicle's initial route is calculated from its origin to its destination. For the ambulance scenario, the destination of the emergency vehicle (i.e. ambulance) is fixed to the hospital's location. For other vehicles, the nodes and vertices of road features are used both for an origin and a destination to build their route. Then, in every time 'tick',[28] each vehicle agent updates its location information and checks whether there is a request/warning from an emergency vehicle or a broken-down vehicle. If there is an emergency situation, a vehicle proceeds to a decision-making process based on its own location and the information transmitted from an emergency vehicle to allow its driver to take action appropriate to the occasion.

[28] Time within an agent simulation environment is regarded as a discrete event whose quantum unit of time is known as a 'tick' (Crooks, 2007). If event `x` and `y` are scheduled at tick `one` and `two` respectively, event `y` will execute after `x`.

Table 6.8 - Vehicle agent's attributes and behaviours

Vehicle class	Attributes	Behaviours	Remarks
Vehicle (all vehicles)	origin, destination, current coordinates, current speed, currentRoad, nextRoad prevJunction, nextJunction, distanceToNextJunction, timeToNextJunction, routeSegments, routeJunctions	travel()	Common attributes and behaviours
Private vehicle		getRequested(), answerBack(), changeVelocityIDM(), changeVelocityJunctionManagement(), stop(), moveOverToTheRightLane()	Reactive behaviours
Emergency vehicle, Broken-down vehicle	distanceRange, timeRange	request(), warning()	Proactive behaviours

6.4 Agent simulation[29]

The previous two sections described vehicle agents' attributes and behaviours as well as their virtual road environment to cover dynamic interactions, mainly communications, among vehicles to share their geospatial information to resolve the emergency situations in the scenarios.

This section outlines several important points that are used to develop agent models and to test them using simulations. Repast Simphony was chosen as the agent modelling and simulation platform. Thus, explanations and descriptions about agent implementation are closely related to Repast Simphony. Amazon Elastic Compute Cloud (for a Windows platform) and a Mac OS X platform are used as the simulation servers.

Section 6.4.1 explains the context and projection, which are agents' container and model space, respectively. Then, Section 6.4.2 outlines descriptions about agent classes and their hierarchy. It covers the setup procedure, which outlines initial parameters and variables for agents and the environment, and the dynamic procedure, which covers the agents' states including their speed and route as well as the agents' behaviours, such as their movements and giving way to an

[29]For the simulation, Repast Simphony 1.2.0 was used, and the workspace (for source code and data) is obtainable at https://goo.gl/aaOxhA. Major parts of the source code are also provided in Appendix D.

ambulance. Finally, Section 6.4.3 shows what kinds of data can be extracted and logged during the simulations.

6.4.1 Agent, context, and projection

In Repast Simphony, contexts and projections are used to organise agents and their space. As described in Section 3.2.3.3, a context acts as a container for agents while a projection represents agents' location and their spatial relations in the context. In the simulation, a context and two projections are used (Table 6.9). Vehicle agents, road elements, and junctions are added into the context. Road elements and junctions are the basic elements used to build the geometry of the road environment. A road environment can be represented as a geographic space and as a network. Therefore, two projections, a GIS topology model space (i.e. roadGeography) and a network topology model space (i.e. roadNetwork), are used in the simulation.

Table 6.9 - Context and projection used in the simulation

Type	Elements

Context	• vein.**Vehicle**
	  o vein.**EmergencyVehicle**
	  o vein.**BrokenVehicle**
	  o vein.**PrivateVehicle**
	• vein.**RoadElement**
	• vein.**Junction**

Projection	• GIS topology model space: **roadGeography** • Network topology model space: **roadNetwork**

The GIS space is needed to allocate vehicle agents' location on the road, and the network space is used to calculate vehicles' shortest route based on the weights (e.g. lengths) of the edges. With the two model spaces (projections), vehicles' movements and their spatial/social relations can be represented. Since each projection offers various methods to query, calculate, and analyse features and objects in the model space, various geographic (spatial) analyses can be performed, for example, spatial overlay and buffer analysis for the geographic projection, and network analysis for the network projection.

6.4.2 Classes and their hierarchy

For the agent simulation, several Java classes are built on top of the Repast Simphony platform. Some Java classes are made to represent vehicle agents and road network, and other classes are designed to initialise parameters, build the context, and set the projection for the agents' environment. These classes can be categorised into four groups, namely, the parameters and simulation initialisation group, the road geography and road network group, the vehicle and route group, and the vehicular communication group (Table 6.10). The first group is composed of the classes that initialise the simulation environments and set the parameters, which can affect the simulation results. The second group contains the classes that generate the road environment for the vehicles to move on. The third group contains the classes for vehicle agents and their route. Last, in the fourth group, there is a communication message class duplicating functionality from the VEIN application ontology. Its properties and methods are logically mapped from properties and query functions/templates of the VEIN application ontology (Table 6.11). Even though the class is not directly integrated with the VEIN application ontology, it reproduces vehicular communications to trigger the vehicle agents' reactive behaviours during the simulations.

Table 6.10 - Class groups and class hierarchy for the agent simulation

Class group	Class and class hierarchy
Parameters and simulation initialisation	• vein.**GlobalVariables** • vein.**VeinContextCreator** (implements repast.simphony.dataLoader.ContextBuilder<T>) • repast.simphony.context.DefaultContext<T> o vein.**ContextOne** • vein.**Infra**
Road geography and network	• vein.**RoadElement** • vein.**RoadEdge** • repast.simphony.space.graph.RepastEdge<T> o vein.**MyRepastEdge**<T> • vein.**Junction**
Vehicle and Route	• vein.**Vehicle** o vein.**EmergencyVehicle** o vein.**BrokenVehicle** o vein. **PrivateVehicle** • vein.**Route** • vein.**MyShortestPath**<T> (implements repast.simphony.space.projection.ProjectionListener<T>)
Vehicular Communication	• vein.**CommunicationMessage**

Table 6.11 – The CommunicationMessage Class extracted from the VEIN application ontology

VEIN application ontology	The `CommunicationMessage` class
geo:where	`double x, y;`
vein:isAtASpeedOf	`double speed;`
vein:previousJunction	`Junction previousJunction;`
vein:comingJunctions	`ArrayList<Junction> comingJunctions;`
vein:nextJunction	`Junction nextJunction;`
vein:currentRoadElement	`RoadEdge currentRoadEdge;`
vein:nextRoadElement	`RoadEdge nextRoadEdge;`
vein:remainingDistanceToNextJunction	`double remainingDistanceToNextJunction;`
vein:remainingTimeToNextJunction	`double remainingTimeToNextJunction;`
vein:updateRelativeLocation	`boolean updateRelativeLocation()`
vein:fnInTheSituationOfEmergencyVehicle	`boolean isInTheSituationOfEmergencyVehicle()`
vein:fnInTheSituationOfVehicleStationary	`boolean isInTheSituationOfVehicleStationary()`

Meanwhile, each class in Table 6.10 can be seen as a part of the modelling procedures (i.e. setup procedure and dynamic procedure). The next two sections describe the classes in the setup procedure to initialise the virtual road environment and vehicle agents' properties, and the dynamic procedure to update vehicles' states (e.g. movement, communication, and reaction), respectively.

6.4.2.1 Setup procedure (initialisation procedure)

In the setup procedure, the variables for vehicle agents and the road environment are initialised. Then, the virtual road environment is built, and vehicle agents are instantiated with their properties such as speed, route, etc.

The `GlobalVariables` class is composed of global parameters that can affect the initial settings of vehicle agents and the road environment. Assigned values of global parameters are set when the simulation is executed from a parameter file in the XML format. The batch mode is used with a parameter file to obtain simulation results from different percentages of communicative vehicles. A parameter file sets the initial values for the vehicle population, the percentage of DSRC vehicles, the DSRC vehicles' communication range, the number of ambulances or broken-down cars, a shape file name for the road environment, etc.

The `VeinContextCreator` class implements the `ContextBuilder` interface to build a context as a container of agents and a projection to set a space for the context. It has a `build(Context<Object> context)`method that plays a role similar to a `main()` function in C or Java programming. This function uses some methods of the `Infra` class to generate a road environment from a shape file and instances of vehicle agents. The European Petroleum Survey Group (EPSG) Projection 27700 (i.e. British National Grid) was used as the projection for the context. The British National Grid is a projected coordinate system, which makes it easier to deal with distance-based calculations for vehicles' movements during the simulation.

All instances for the road environment and vehicle agents are built in the `Infra` class. This class generates road elements, a road network, and vehicle agents, and sets the vehicles' properties, such as their origin, destination, and route. The `Infra` class has methods to create and set instances of the road element, the road network, and the vehicles. It also has some methods to get/find agents from the context. To represent the road geometry, each instance of the `RoadElement` class is generated from each feature (polyline) of the road shapefile, which is extracted from the ITN layer. After loading the shapefile, the route of an ambulance or a broken vehicle is defined, and then the road elements that are located away from the route are removed by buffering analysis to simplify the virtual road environment. Instances of the `RoadElement` class have properties for one-way roads and some restrictions, and these properties are used to generate a road network, which has realistic routes and road access. The road network is composed of edges (instances of the `MyRepastEdge` class) and junctions (instances of the `Junction` class). A feature (polyline) of a road element has two nodes and additional vertices to express the road element's geometry information. Junctions are generated from nodes (two end points) of road element features, and each edge of the network is created by using two junctions and the length of a road element. To represent a one-way street, directed edges are used for the road network, so a two-way street has two road edges to allow vehicles to travel in both directions.

The `Vehicle` class defines the properties and methods for vehicles, and it is the super class of the `EmergencyVehicle` class, the `BrokenVehicle` class, and the `PrivateVehicle` class. A vehicle's origin, destination, speed, and route are initialised in the setup procedure. The `Route` class generates a vehicle's moving track on the road with reference to the road geography and road network to set and update the vehicle's movement and journey during the simulation. For a vehicle's route, its origin junction and destination junction are chosen randomly, and then the `MyShortestPath` class examines whether there is a possible shortest path between the origin and the destination. Dijkstra's algorithm (Dijkstra, 1959) is used for the shortest path, which is provided by the Java Universal Network/Graph Framework (JUNG) library, as a part of Repast Simphony. If a vehicle's origin and destination are given too close to each other and their Euclidean distance is less than 1 km, either the origin or the destination will be reassigned until the vehicle gets a proper path. The `RoadEdge` class is designed for a vehicle's route, which is a list of directed road elements from the origin to the destination.

6.4.2.2 Dynamic procedure

During the simulation, there are three dynamic methods for vehicles' movements, communications, and reactions. These methods are included in the `step()` method of the `Vehicle` class, and they are checked and executed in each 'tick' for vehicles to take action based on their current location. First, the `travel()` method of the `Vehicle` class represents the vehicles' movements. A vehicle's travel distance per unit time is dependent on the vehicle's speed and the duration of the unit time, and a global variable `TRAVEL_PER_TURN` is used for vehicles to convert their speed to distance in every time 'tick' in order to update the vehicle's location during the simulation. To obtain a vehicle's next location in each 'tick', the vehicle needs to have a list of vertices of road edges in the route, and its next location can be a vertex, a node, or somewhere between them. Second, the `EmergencyVehicle` class and `BrokenVehicle` class, which inherit the `Vehicle` class, have additional properties and methods to send and receive communication messages. There are two methods; `request()` is used to make requests and `getAnswerback()` is used to obtain responses from affected vehicles. Third, vehicles in the vicinity of an emergency vehicle or a broken-down vehicle need to react properly. There are two methods; these are `stop()` to give way to the ambulance and `moveOverToTheRightLane()` to avoid the broken-down vehicle.

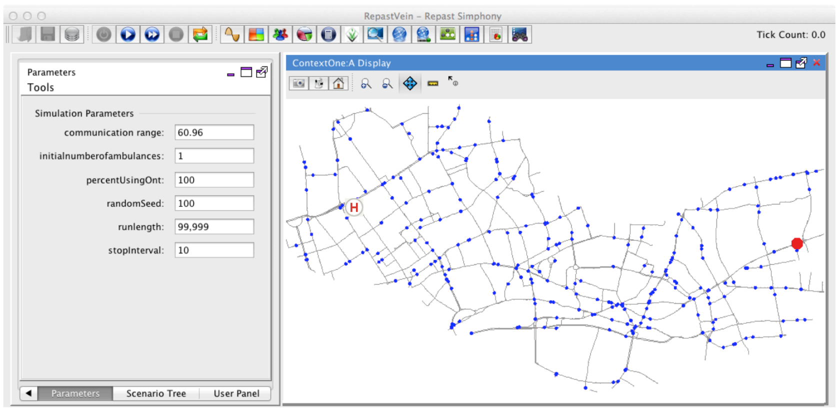
Each vehicle has its own instance of the `CommunicationMessage` class, and its `updateRelativeLocation()` method is used to update the vehicle's current relative location. When a vehicle get a request from an emergency vehicle or a broken-down vehicle, either of the `InTheSituationOfEmergencyVehicle()` method or the `isInTheSituationOfVehicleStationary()` method is used to check if the vehicle needs to take an action or not. This class is also a container to share a vehicle's partial route and its relative location based on their remaining distance and remaining time to a junction or a road's node. To describe a partial route, some properties for the previous and next junctions, current and next road edges, and upcoming junctions are defined. In addition, there are some methods to obtain the remaining distance and time to the next junction from a vehicle's current location to calculate a vehicle's relative location.

The `getRemainingDistanceToNextJunction()` method translates a vehicle's absolute location (two-dimensional coordinates) into a numeric value, which represents the distance from the vehicle's location to the next junction to simplify the communication contents and the decision-making process. Vehicles have a different speed and different remaining time to the junction even if two vehicles have the same remaining distance to the same next junction. So, there is the `getRemainingTimeToNextJunction()` method to allow a vehicle to obtain the relative time to its next junction. In this way, a vehicle's absolute location can be represented relatively with reference to a junction. If several vehicles are heading towards the same junction, the vehicle's location can be represented relatively.

6.4.3 Simulation outputs

Based on initial parameters and randomised variables, the simulation runs may give different results. To obtain and analyse the data and statistics of the simulation results, some data are extracted and logged as outputs. Repast Simphony contains a package called `repast.simphony.data.logging.outputter` for data logging in various formats and time durations. For the simulation results of the ambulance scenario and the breakdown scenario, the ambulance and the broken-down vehicle collect some data related to themselves and to the vehicles around them per every time 'tick'.

Figure 6.8 shows an example of the ambulance scenario where a rapidly moving ambulance is heading towards a hospital. In each time 'tick' (0.1 sec), the locations of the ambulance and the other vehicles are updated, and the ambulance's communication targets change constantly during the journey towards the hospital. The ambulance is shown as a big red point, while the vehicles around it are shown as smaller blue points. The ambulance shares its location and partial route information in real-time, and the effect of vehicular communications on the traffic efficiency in an emergency situation is measured during the simulation to support the transition towards ITS.


RepastVein – Repast Simphony

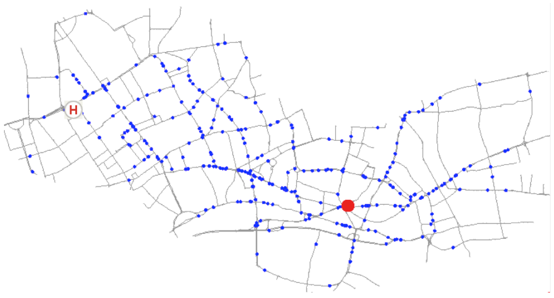
Tick Count: 0.0

Parameters
Tools

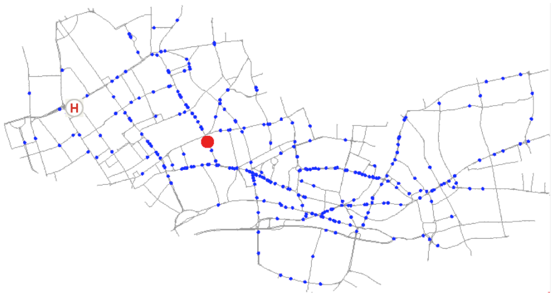
Simulation Parameters

communication range: 60.96
initialnumberofambulances: 1
percentUsingOnt: 100
randomSeed: 100
runlength: 99,999
stopInterval: 10

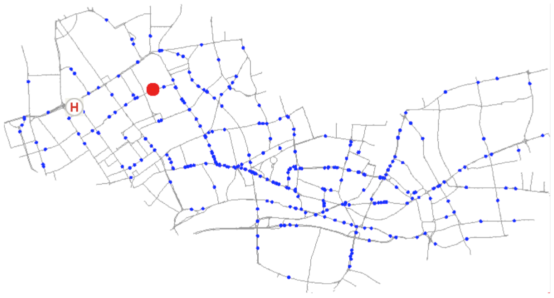
(a) Initial allocation of vehicle agents



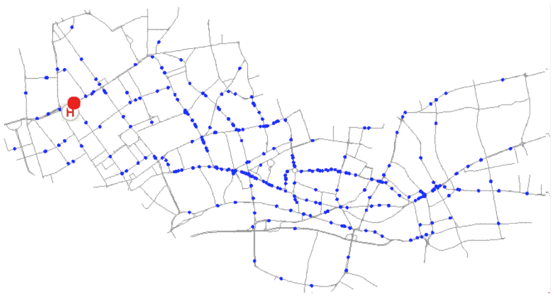
(b) After 1000 ticks



(c) After 2000 ticks



(d) After 2500 ticks



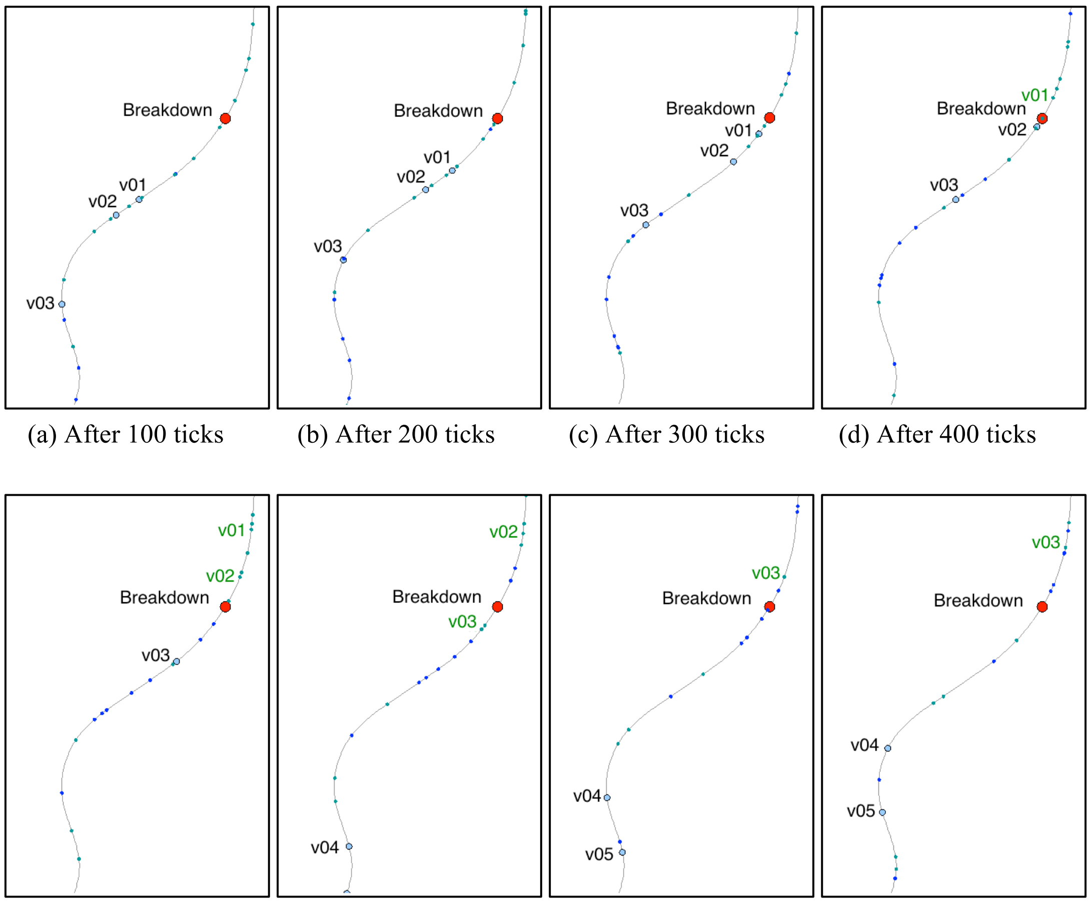
(e) After 3000 ticks

Figure 6.8 - An example of simulation for the ambulance scenario around University College Hospital[30]

Figure 6.9 shows an example of the breakdown scenario where rapidly moving vehicles are heading towards a breakdown spot. In each time 'tick', the locations of the vehicles are updated, and the broken-down vehicle's communication targets change constantly. The broken-down vehicle is shown as a red point, while vehicles that need to change lanes because of the breakdown are described as sky-blue points with labels (i.e. v01, v02, v03, etc.). The broken-

[30]A dynamic example of the ambulance simulation is obtainable at https://youtu.be/-vgDR77Rels.

down vehicle shares its location and partial route information via vehicular communication, and the effect of vehicular communications on the traffic efficiency in the breakdown situation is measured during the simulation.



(a) After 100 ticks (b) After 200 ticks (c) After 300 ticks (d) After 400 ticks

(e) After 500 ticks (f) After 600 ticks (g) After 700 ticks (h) After 800 ticks

Figure 6.9 - An example of simulation for the breakdown scenario on a motorway[31]

Table 6.12 shows the list of the data logged in every time 'tick' in the simulation runs. The log file can be split into three parts. The first part stores the emergency vehicle's travelling in every 'tick', the second part saves the parameter values of the simulation, and the third part keeps the history of the number of vehicles that receive the request from the emergency vehicle (i.e. the ambulance or the broken-down vehicle) as well as the number of vehicles that are affected/unaffected by the ambulance or the broken-down vehicle. In the ambulance situation, the vehicles that slowed down, stopped and gave way to the ambulance are considered as the

[31]A dynamic example of the breakdown simulation is obtainable at https://youtu.be/PK8H_OELnSI.

affected vehicle. In the breakdown situation, the affected vehicles mean the vehicles that could not change lanes because of through traffic on the right lane they wish to enter so that they are slowed down and stopped behind the breakdown until they find an opportunity to change lanes.

During the simulation, the effect of the communications (sharing an emergency vehicle's presence, relative location, partial route, and emergency rules) on the traffic flow is measured. If a vehicle is not disturbed by the emergency vehicle (i.e. the ambulance or the broken-down vehicle) even though it is inside the effect zone of the emergency vehicle, the vehicle is counted as an unaffected vehicle. In the `PrivateVehicle` class, static variables are used to obtain the number of affected vehicles and unaffected vehicles by the emergency vehicle's presence.

Table 6.12 - The list of variables in the log file during the simulation

Variable	Description
Tick()	Accumulated time (number)
this.getName()	The message sender vehicle's name
this.getSpeed()	The message sender vehicle's speed
this.getCurrentCoordinateX()	The vehicle's current x coordinate
this.getCurrentCoordinateY()	The vehicle's current y coordinate
this.getCurrentRoadFID()	Current road ID that the vehicle is on
this.getNextJunctionID()	Next road ID of the vehicle will be on
this.getRemainDistanceToNextJunction()	The vehicle's remained distance to the next junction
this.getDTravelTime()	The vehicle's accumulated travel time (sec)
this.isArrived()	True when the vehicle arrives at the destination
GlobalVariables.iRondomSeed	Random seed value for random number generator
GlobalVariables.iPercentUsingOnt	Percentage of the vehicles that have an OBE
GlobalVariables.numVehicle	Total number of vehicles except the message sender
GlobalVariables.numEmergencyVehicle	The number of message sender
GlobalVariables.commrange	Communication coverage distance (metre)
this.getNumOfVehicleRequested()	The total number of vehicles that receive the message
this.getNumOfDSRCVehicleRequested	The number of DSRC vehicles that receive the message
this.getNumOfNonDSRCVehicleRequested()	The number of non-DSRC vehicles
this.getNumOfVehicleAffected()	The total number of affected recipient vehicles
this.getNumOfDSRCVehicleAffected()	The number of affected DSRC vehicles
this.getNumOfNonDSRCVehicleAffected()	The number of affected non-DSRC vehicles
this.getNumOfVehicleUnaffected()	The total number of unaffected recipient vehicles
this.getNumOfDSRCVehicleUnaffected()	The number of unaffected DSRC vehicles
this.getNumOfNonDSRCVehicleUnaffected()	The number of unaffected non-DSRC vehicles

This section demonstrated the classes and their hierarchy and showed how this information is used to implement the simulation based on the Repast Simphony platform. Each class is illustrated with its methods in the setup procedure and the dynamic procedure, which initialise the simulation environment and update the vehicle agents' attributes and states in every 'tick'. It also outlined which variables are extracted and logged during the simulations as simulate outputs.

The next section explains the process used to verify and validate the simulation logic and justifies the sample size of the simulation (i.e. the number of simulation runs) to get statistical results. This process is a preliminary process to check the simulation logic and model, so it has to be done before the execution of the simulation.

6.5 Pilot simulation

Before the main simulation is run, the simulation logic and model have to be justified to check whether they represent the situations of the scenarios. In addition, a statistical explanation is required to decide how many simulations are required. Section 6.5.1 describes the simulation verification and validation process, and Section 6.5.2 explains how the number of simulation runs is decided in order to statistically analyse the results.

6.5.1 Verification and Validation

The verification process is concerned with the simulation logic and whether it is working as expected, while the validation process is concerned with whether the simulation models the real system properly (Abdou et al., 2012). Since vehicles travel on a road, vehicle agents' movements and reactions are verified by visual display. To verify the vehicle agents' reactions in accordance with the emergency rules in the vicinity of an ambulance and a breakdown, a set of test cases is used in a similar way to the ontology validation described in Section 5.5. The test set is made with a small size road network and small number of vehicles to verify the behavioural rules, which are described in the ontology model, and to check whether vehicles move and react as designed. The test set represents an extreme situation (i.e. a much smaller number of vehicles and a smaller road network), so that the vehicles' movements and interactions in the simulation can be easily predicted for the verification.

For modelling and simulations, there is a common validation pattern, which is to compare the output of the simulation with real data collected about the targeted 'real system' (Abdou et al., 2012). The main objective of this research is to explain possible interactions among vehicles based on vehicular communications. For the vehicular mobility model of the simulation, synthetic models such as a car-following model (IDM) and a simple FIFO-based junction management model, and a communication-based behaviour model are adopted and implemented. Even though the mobility model in this simulation is not based on real surveys or traces, most of the VANET simulations use a synthetic mobility model, which is based on mathematical equations, to produce a realistic vehicular mobility (Al-Sultan et al., 2014). The mobility model of this simulation is acceptable in this regard. So, the verification of the vehicle agents' reactions can be seen as a predictive validation without any replication of an existing situation in a specific region, and the model is acceptable in this regard. However, this simulation model can be extended to replicate existing traffic situations with real data in the future.

6.5.2 How many simulation runs?

To analyse the simulation results, it is necessary to have statistical distributions over many simulations runs. According to Helbing and Balietti (2012), at least 100 simulation runs are required for statistical analysis. In order to decide the size of simulation runs, traditional statistical approaches for determining sample size, and various precision levels are considered (Table 6.13).

Basically, three parameters (i.e. precision, confidence level, and variability) are needed to yield a proper sample size from large populations that follow a normal distribution (Israel, 2012; Miaoulis and Michener, 1976). Table 6.13 presents approaches for determining sample size with a 95% confidence level and the maximum degree of variability based on the assumption that the measured attributes of the sampling are dichotomous (Israel, 2012). Since there are several random parameters such as vehicles' initial locations and their routes in the simulations, there could be a countless number of simulation cases. So, in this calculation, it is considered that N is bigger than 100,000. The confidence level can be set at 90%, 95%, or 99%, but a 95% confidence level is used for determining the number of simulating runs since it has been generally stated in applied practice (Millner et al., 2006; Assessment Operation Group, 2012).

Table 6.13 - Different approaches for determining sample size

Approach	Description or equation	Precision level (e)	Confidence level (Z)	Degree of variability (P)	Sample size (n)
Using a published table (Yamane, 1967)	$n = \frac{N}{1 + N(e)^2}$ (size of population N > 100,000)	±10% (0.1)	95% (1.96)	Maximum (0.5)	100
		±7% (0.07)			204
		±5% (0.05)			400
		±3% (0.03)			1,111
Formula for proportions (Cochran, 1963)	$n = \frac{Z^2 p\,(1-p)}{e^2}$	±10% (0.1)	95% (1.96)	Maximum (0.5)	96.04
		±7% (0.07)			196.00
		±5% (0.05)			384.16
		±3% (0.03)			1067.11

Even though the measured attributes (e.g. delayed time of vehicles around an ambulance or a broken-down vehicle) of the simulations are continuous and cannot be grouped into two categories, the approaches based on proportions with the maximum variability, described in Table 6.13, are adopted for determining the sample size of the simulation runs. Yamane's table produces more conservative (larger) sample sizes than does Cochran's formula, so sample sizes from Yamane's table are used here.

When assuming a 95% confidence level and maximum variability, the sample size is dependent upon the precision level. To choose a proper sample size of the simulation runs, sample sizes from four different precisions (i.e. ±3%, ±5%, ±7%, and ±10%) are considered. Preliminary simulations are executed with four different runs for the ambulance scenario, and their results are analysed. The number of vehicles affected by the ambulance, and the ambulance's travel time are reviewed separately by the analysis of variance (ANOVA) for a single factor. The null hypothesis was that the means and variances of the four groups are similar, and a 95% confidence level (significance level a = 0.05) was assumed. The results of ANOVA show that the p value is bigger than the significance level a (0.05), and the F calculated value is smaller than the F critical value for each simulation, so the null hypothesis is accepted (Table 6.14). It justifies that four different groups, which represent four different numbers of simulation runs, have similar means and variances. Therefore, 100 simulation runs are executed and analysed for each situation. All the statistical summaries of variance analysis for these four groups are presented in Appendix C.

Table 6.14 - ANOVA of the four groups to decide the number of simulation runs

Single factor	*P-value*	*a*	*F*	*F crit*	Comparison	Result
The number of vehicles affected by ambulance	0.526	0.05	0.744	2.610	*P value > a, F < F crit*	Accept Null Hypothesis
The total affected time of vehicles by ambulance	0.614	0.05	0.602	2.610	*P value > a, F < F crit*	Accept Null Hypothesis
Ambulance' travel time	0.567	0.05	0.676	2.610	*P value > a, F < F crit*	Accept Null Hypothesis

6.6 Simulation results

This section presents the simulation results of two scenarios, namely, the ambulance scenario and the breakdown scenario. As mentioned in Sections 6.3.1 and 6.5.2, the simulations are designed to provide 100 result sets of 5 different communication environments. Each set uses the same settings except for a different portion of OBE-implanted vehicles (DSRC-equipped vehicle) from 0% to 100% (Table 6.15). A situation that has 0% of OBE-implanted vehicles represents the 'conventional'[32] approach, while a situation with 100% of OBE-implanted vehicles represents the 'vehicular communication' approach. In the simulation, an OBE-implanted ambulance or broken-down vehicle only sends the communication messages, and the OBE-implanted vehicles around it receive the messages and take actions to resolve the situation. Therefore, it is implied that the effect of DSRC-equipped vehicles represents the effect of the communication messages on the traffic flow in emergency situations.

For each of the three different cases of the ambulance scenario, as well as for the breakdown scenario on the motorway, 100 sets of simulation results are generated as described in the simulation output section (i.e. Section 6.4.3). Each set has randomised initial states (e.g. vehicles' routes and a road network). The number of vehicles and the size of the road network can have an influence on the simulation results, so these are kept consistent for each set of simulations. The road network for each set of simulations is generated in a small area along the ambulance's route or the breakdown spot by using buffering, and the number of vehicles is calculated based on the length of a road network of each simulation. In the simulation, vehicles'

[32] By conventional approach we mean traditional transport interactions on the road without vehicular communications, e.g. an ambulance's siren, drivers' visual scanning behaviour, etc.

preferred speed are set based the free-flow speed[33], but they could not travel at free-flow speed when approaching the lead vehicle or road junctions.

Table 6.15 - Simulation summary

Scenario	Road network	The number of Vehicles	Hybrid Environment for Vehicular Communications
Ambulance Scenario (three different hospital locations)	Buffered and minimised based on ambulance's route or breakdown spot	Constant traffic flow[34] based on the total length of a road network	Set of five different communication environments (from 0% of OBE-implanted vehicles to 100% by 25%)
Breakdown Scenario on the Motorway			

We now turn to outline the simulation results for the ambulance scenario and the breakdown scenario. Each section presents the simulation results of each scenario, and it contains a simple description of the simulation and the specific meaning of the efficiency in the simulation. It also provides the effect of vehicular communications on the efficiency trends of traffic in emergency situations, and statistical analysis of the efficiency trends.

6.6.1 Simulation results of the ambulance scenario

The simulation for the ambulance scenario is designed to provide an environment in which an ambulance shares information about its presence and its movements via its siren or vehicular communication. In the scenario, vehicular communications are used to state an ambulance's priority to neighbouring vehicles. An ambulance's siren as well as vehicular communication is used to help the ambulance to reach the hospital as quickly as possible. In either case (i.e. the use of siren or vehicular communications), the ambulance can travel without slowing down if all cars from all directions pull over at once. Therefore, to compare the conventional approach (i.e. ambulance's siren) and the vehicular communication approach in the situation of the ambulance scenario, a simple comparison, such as an ambulance's travel time to the hospital, is inadequate. Nevertheless, it is fair to say that the ambulance can share more detailed information about its presence and movement beforehand via vehicular communications. The communication provides neighbouring vehicles with enough time to respond and give way to the ambulance.

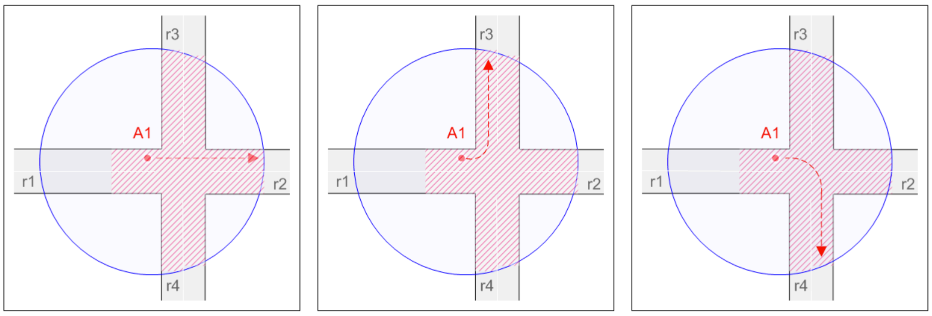
[33] Each vehicle's preferred speed is randomly set by using the free-flow car speed range in Table 6.2a for the ambulance scenario and Table 6.2d for the breakdown scenario.
[34] A quarter of the maximum number of free-flow vehicles on a road network is used for each simulation. The maximum number of free-flow vehicles is calculated based on the assumption that vehicles travel at the speed limit and keep the two-second rule.

Even though an ambulance's request zone is designed to be the same[35] during the simulation, there is a difference in the ambulance's effect zone between the 0% DSRC-equipped vehicle group and the 100% DSRC-equipped vehicle group (Figure 6.10). The 0% DSRC-equipped vehicle group represents the conventional method for an ambulance to request priority by using its siren, and the effect zone of the ambulance's siren (i.e. the patterned area of Figure 6.10a) is almost the same as its request zone. When the drivers of vehicles hear the ambulance's siren, on most occasions, they will be affected by the ambulance to give it a clear lane to pass (or give it more room to travel by stopping or pulling over if the roads are crowded). An ambulance's goal is to bypass traffic at the highest possible speed to save a life, and it can be done safely and efficiently when the nearby vehicles make way for the ambulance. Meanwhile, in the 100% DSRC-equipped vehicle group, the ambulance broadcasts its own location and route via communication messages, so its effect zone (i.e. the patterned area of Figure 6.10b) can vary depending on its route. If a recipient vehicle is informed that its route is not overlapping with the ambulance's route, the recipient vehicle will not need to give way to the ambulance and may not be affected by the ambulance's presence even though it is in the ambulance's broadcast range.

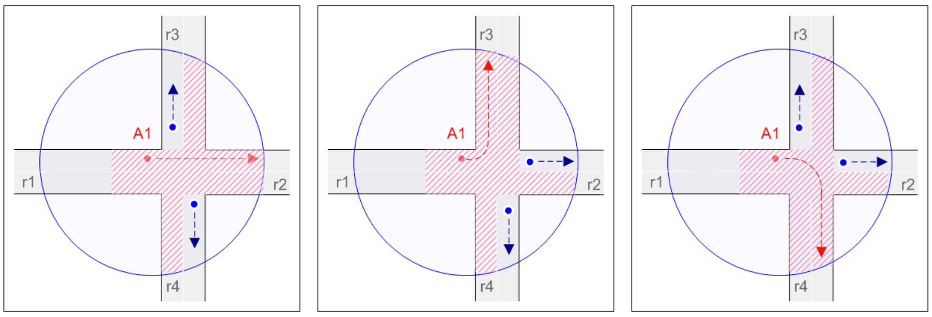
With this simulation environment, it is expected that the total number of affected vehicles may decrease with the numerical increment of DSRC-equipped vehicles because the effect zone of an ambulance's DSRC is relatively smaller than its siren's effect zone. In other words, the total number of unaffected vehicles (i.e. the total number of requested vehicles – the total number of affected vehicles) will increase with the increase in the number of DSRC-equipped vehicles, so that undesirable delays of neighbouring vehicles will be minimised. Therefore, in the

[35] The effective distance of siren penetration through closed windows of moving vehicles can be dramatically short. To remedy this shortcoming, low-frequency siren has been developed. According to the Emergency Medical Services Authority (EMSA, 2009) in Oklahoma, the howler sirens emit low-frequency tones that cause objects within 200 feet (i.e. 60.96 metres) to reverberate, and catch drivers' attention much more quickly. The low-frequency sound waves penetrating solid materials allowing vehicle operators to feel the sound waves. The Howler is not a replacement to the vehicle's primary siren. Howler low frequency tone siren is an effective added layer of warning for intersections and high risk areas. So, in the simulation, an ambulance's request zone was set to 60.96 meters by referring to the effect range of the new howler sirens of ambulances. For the DSRC-equipped vehicles, they may get the message from an ambulance in advance and take more time to be ready for the situation, but they cannot give way for the ambulance in advance if they are too far from each other. A DSRC-equipped vehicle's time window for making way for an ambulance will be similar with a non-DSRC vehicles if the non-DSRC vehicle already noticed the coming ambulance. The ambulance scenario was focused to measure the effects of the communication sharing precise info including its partial route (Figure 6.10), minimising the disturbances by the ambulance. So, the effect zone was set to 60.96 m both for the siren and the DSRC.

simulations of the ambulance scenario, this tendency may become more pronounced as the number of DSRC-equipped vehicles increases, and it can be considered that the efficiency of the whole system (i.e. the efficiency of traffic in an ambulance situation) is increased.



(a) Effect zone of the ambulance `A1`'s siren (the 0% DSRC-equipped vehicle group)

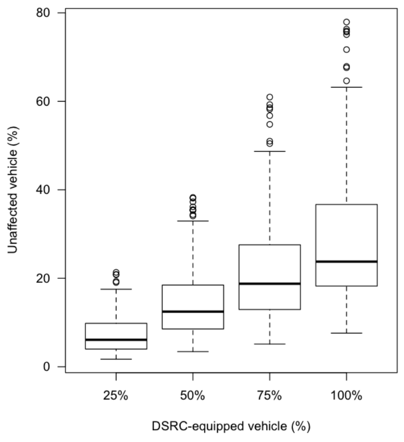


(b) Maximum effect zone of the ambulance `A1`'s communication messages (the 100% DSRC-equipped vehicle group)

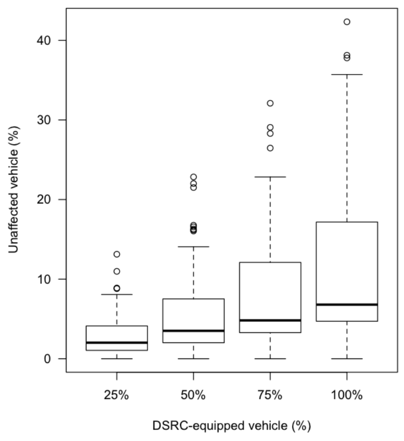
Figure 6.10 - Different effect zones of an ambulance

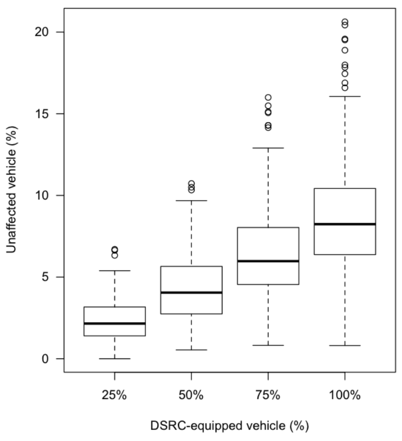
The simulation results reveal an increasing trend of unaffected vehicles corresponding to the increase in the number of DSRC-equipped vehicles (Figure 6.11). In the simulations, vehicular communications play a positive role in minimising the number of vehicles that are affected by the ambulance's presence. In other words, the vehicular communications (replicating the VEIN application ontology) are more efficient than an ambulance's siren when it comes to reducing the undesirable delays of neighbouring vehicles.

However, as demonstrated by the visual patterns in Figure 6.11, there are variations in the degree of the increasing trends of unaffected vehicles. Some cases show strongly increasing trends of unaffected vehicles. In other cases, the increasing trends of unaffected vehicles are relatively weaker. The various degrees of the increasing tendency in the efficiency trends may depend on the road geography and vehicles' relative locations and routes, which were randomly set for each simulation. In most cases, the outliers, which are beyond the upper quartile in the boxplots of Figure 6.11, are appeared when the ambulance's route in a simulation contains roads with reservation (i.e. dual carriageways). Further research is required to study whether road geography affects the degree of the increasing trend of the percentage of unaffected vehicles in an emergency situation, but it is beyond the scope of this research.



(a) University College Hospital





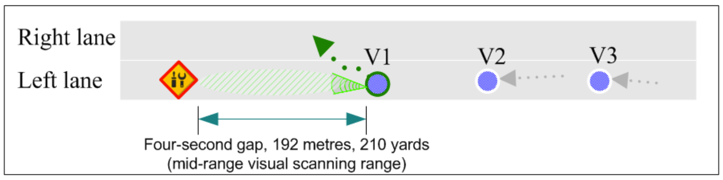
(b) Kingston Hospital (c) Princess Royal University Hospital

Figure 6.11 - Trends of the percentages of unaffected vehicles in eleven vehicle groups

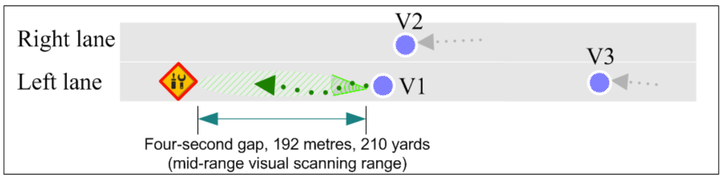
6.6.2 Simulation results of the breakdown scenario

The previous section discussed the simulation results of the ambulance scenario, which showed the effect of vehicular communications on traffic efficiency in an emergency situation by focusing on minimising the effect zone of an ambulance. This section outlines the simulation results of the breakdown scenario on the motorway. It presents the effect of vehicular communications in a breakdown situation by focusing on extending the request range for a breakdown spot. Consequently, the simulation for the breakdown scenario is designed to provide an environment in which a broken-down car broadcasts its presence and warning messages far beyond other drivers' visual scanning range. These broadcasts give (the drivers of) the vehicles heading towards the breakdown spot more time to reduce speed or change lanes safely.

Even though the drivers' response times for a sudden traffic event/accident may vary, the effect of the traffic event can be minimised if the drivers of the vehicles heading towards the breakdown spot are informed about the situation beforehand and have enough time to react. Therefore, a breakdown's request zone is designed to be different from that of an ambulance. In the 0% DSRC-equipped vehicle group, a four-second gap[36] at the speed of 70 mph (i.e. 192 metres, 210 yards) is regarded as drivers' average response range (Figure 6.12). In contrast, in the 100% DSRC-equipped vehicle group, a breakdown's request zone is extended to a 16-second gap (i.e. 768 metres, 840 yards, twelve-second gap + four-second gap) via vehicular communications (Figure 6.13). The safe time gaps for overtaking the breakdown spot for moving vehicles and vehicles that have stopped are set as four seconds and six seconds, respectively.


Right lane
Left lane
V1
V2
V3
Four-second gap, 192 metres, 210 yards
(mid-range visual scanning range)

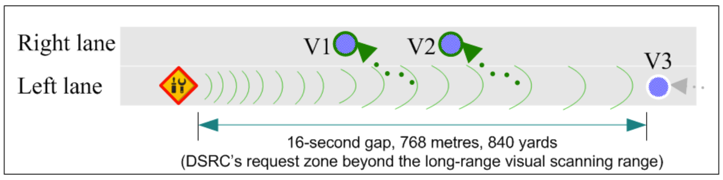
(a) If safe, `V1` moves over to the right lane


Right lane
Left lane
V2
V1
V3
Four-second gap, 192 metres, 210 yards
(mid-range visual scanning range)

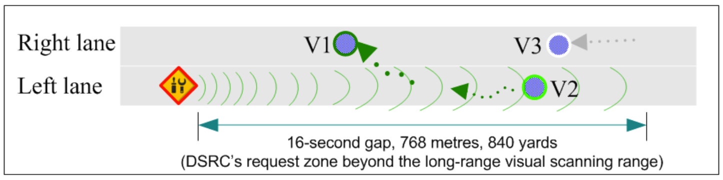
(b) If not safe, `V1` slows down or stops, and then finds a chance to move over

Figure 6.12 - Request zone of a breakdown in the 0% DSRC-equipped vehicle group

[36] The Automobile Association (AA) recommends that a driver on a motorway needs to anticipate what is coming next by sweeping the road ahead visually (look two seconds ahead, four seconds ahead, and twelve seconds ahead). So, a two-second gap, four-second gap, and twelve-second gap can be considered as a short-range, mid-range, and long-range visual scanning distance, respectively. In this simulation, it is considered that four seconds will give drivers time to react easily in a relaxed way. Meanwhile, a 16-second gap is regarded as far long-range distance, which is beyond drivers' visual scanning range.


Right lane
Left lane
V1
V2
V3
16-second gap, 768 metres, 840 yards
(DSRC's request zone beyond the long-range visual scanning range)

(a) If safe, recipient vehicles move over to the right lane


Right lane
Left lane
V1
V3
V2
16-second gap, 768 metres, 840 yards
(DSRC's request zone beyond the long-range visual scanning range)

(b) Even though V2 cannot move over now, it has more time to find a chance to move over

Figure 6.13 - Request zone of a breakdown in the 100% DSRC-equipped vehicle group

With this simulation environment, it is expected that the total number of vehicles that overtake a broken-down car safely without slowing down may increase with the numerical increment of DSRC-equipped vehicles. This is because the request range of the broken-down car in the 100% DSRC-equipped vehicle group is relatively longer than that in the 0% DSRC-equipped vehicle group. A longer request range means that drivers of the vehicles heading towards the breakdown will be warned in advance and so will have more time to move over to the right-hand lanes without slowing down and stopping. Therefore, in the simulations of the breakdown scenario, the effect of the breakdown will decrease as the number of DSRC-equipped vehicles increases. The simulation results show an increasing trend of unaffected vehicles corresponding to the increase in the number of DSRC-equipped vehicles (Figure 6.14).

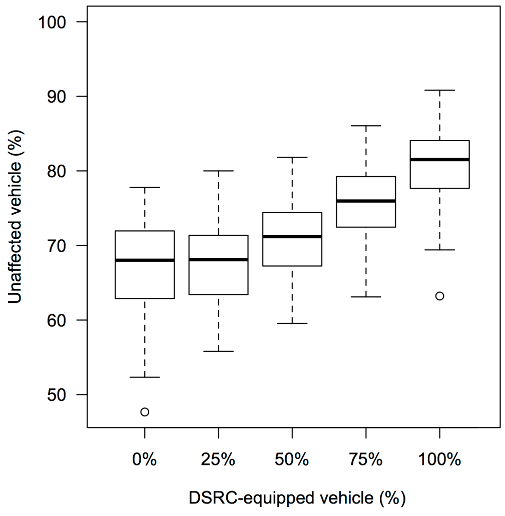


Figure 6.14 - Trends of the percentage of unaffected vehicles in five vehicle groups

The Cullen and Frey graph (Figure 6.15) and the Chi-squared p-value (i.e. 0.18 > 0.05) from the probability distribution, which is created by fitting the normal distribution to the observations (i.e. the percentages of unaffected vehicles), indicate that the observations are from a normal distribution.

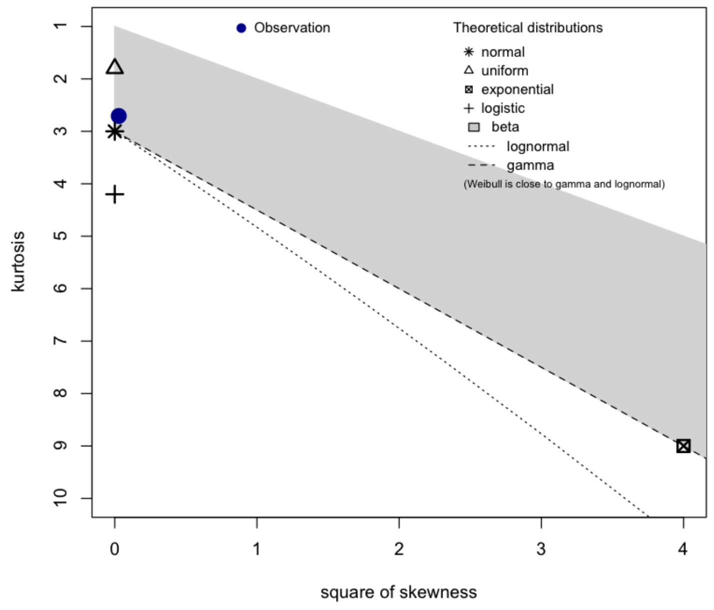
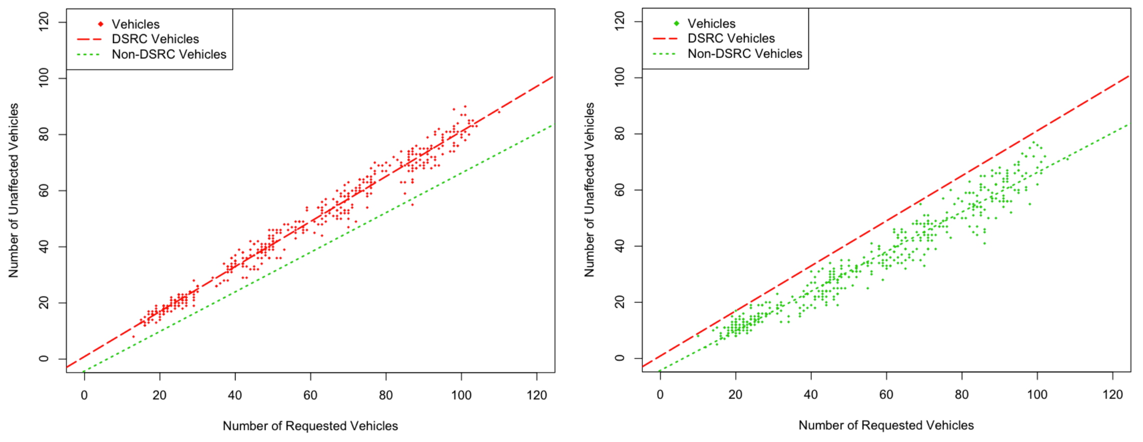


Figure 6.15 - Cullen and Frey graph of the observations (i.e. the percentages of unaffected vehicles)

The proportional relations between the number of unaffected vehicles and the number of requested vehicles are examined in detail. In order to test the proportional relations between the number of unaffected vehicles and the number of requested vehicles, the vehicles are divided into two groups, namely, the DSRC-equipped vehicle group and the Non-DSRC vehicle group. For each vehicle group, a linear regression analysis is performed. The linear regression results show that the increase of unaffected vehicles occurs due to the effect of the DSRC vehicles (Figure 6.16 and Table 6.16). The regression equation of the DSRC-equipped vehicle group has a higher coefficient (i.e. 0.80) and R-squared value (i.e. 0.98). It indicates that the regression model of the DSRC-equipped vehicle group is more significant statistically, and for every additional requested vehicle, you can expect the number of unaffected vehicles to increase by an average of 0.80.



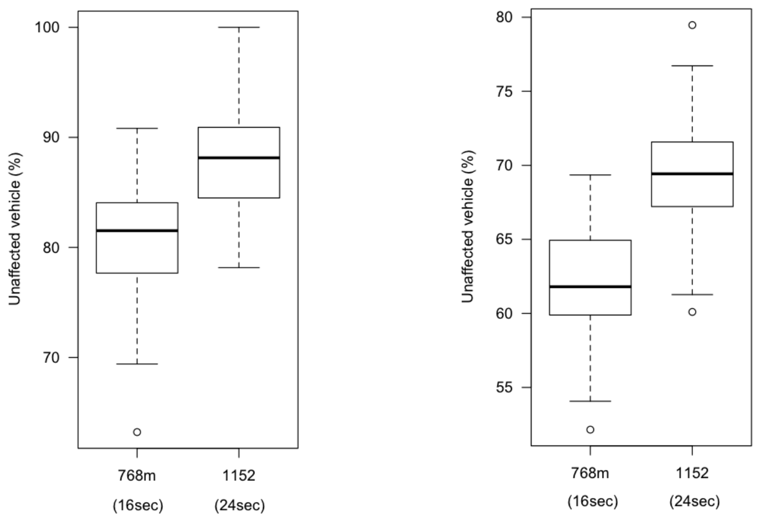
(a) DSRC-equipped vehicles only (b) Non-DSRC vehicles only

Figure 6.16 - Scatterplots and regressions between the number of requested vehicles and the number of unaffected vehicles

Table 6.16 - Coefficients of linear regressions of unaffected vehicles from the breakdown

Vehicle Group	Coefficients				
		Estimate	Std. Error	t value	Pr(>|t|)
DSRC-equipped Vehicles only	intercept	0.9096	0.3860	2.356	0.0189
	x-variable	0.8024	0.0061	131.300	< 2e-16
	Residual standard error: 3.213 on 398 degrees of freedom Multiple R-squared: 0.9774, Adjusted R-squared: 0.9774 F-statistic: 1.724e+04 on 1 and 398 DF, p-value: < 2.2e-16				
Non-DSRC Vehicles only Hospital	intercept	-4.2522	0.5218	-8.149	4.79e-15
	x-variable	0.705515	0.008467	83.328	< 2e-16
	Residual standard error: 4.364 on 398 degrees of freedom Multiple R-squared: 0.9458, Adjusted R-squared: 0.9457 F-statistic: 6944 on 1 and 398 DF, p-value: < 2.2e-16				

In a DSRC-only environment, additional simulation with two different DSRC ranges (768m range versus 1152m range) is also performed to test the impact of DSRC-range changes. For the further extended DSRC range (1152m), two times of the long-range visual scanning distance (two times of twelve-second gap) is used. With the additional extended DSRC range (1152m), vehicles approaching the breakdown could change lanes as farther away as possible, so the simulation result shows a higher number of unaffected vehicles with 1152m communication range than with 768m range (Figure 6.17).



(a) Designed traffic density (b) Two times of designed traffic density

Figure 6.17 - The percentage of unaffected vehicles in two different DSRC ranges

6.7 Summary and discussion

This chapter outlined the process of the design and implementation of vehicle agents and their road environments based on the Repast Simphony platform. Section 6.2 summarised the traffic statistics and the GIS dataset that can describe vehicle agents' movements and the road environment, respectively. Section 6.3 outlined the conceptual modelling for the simulation, and Section 6.4 explained the implementation of the simulation. The preliminary simulation and the statistical analysis of the simulation results were presented in Section 6.5 and Section 6.6, respectively.

First, Section 6.2 showed which information could be used to represent vehicle agents' internal states and road geometry and network. A vehicle agent's belief-desire-intention (BDI) architecture could be simplified into the vehicle's attributes and methods in the simulation, so the object-oriented approach of the agent in the Repast Simphony was adopted without extension to represent vehicle's characteristics. For the vehicles' movements, it seems that using traffic measurements leads to a more realistic simulation because drivers in a free traffic flow often break the regulations and ignore recommendations. For the road environment, route information for public transport (e.g. bus lane and their operating time) and Urban Paths for pedestrians are not considered because considering multi-modal travel is beyond the scope of this research. Thus, the Road Network theme and limited RRI of the ITN layer are used to build a simple directed network describing one-way streets and turning restrictions.

Second, in Section 6.3, the purpose of the simulation and expected results were presented to clarify the reason for the modelling process. Then, two different datasets of road network were extracted from the ITN layer and stored as shapefiles. Vehicle agents' general properties and methods are also designed during this process. In order to focus on vehicular communications, the model assumed that vehicle agents are the only agents, and it differs from other agent simulations of emergent phenomena. The shapefiles and the conceptual model of vehicle agents are independent of the Repast Simphony platform, so they can be used for other agent modelling and simulation platform.

Third, Section 6.4 showed the implementation of the agent simulation on the Repast Simphony platform. It focused on describing the class hierarchy in the setup procedure and the dynamic procedure. When considering a simulation environment with these two procedures, the properties and methods of each class could be easily designed and built. Variables representing the simulation outputs are also collected in every 'tick' for the statistical analysis.

Fourth, Section 6.5 demonstrated a preliminary simulation with two different processes. First, the simulation logic of vehicles' movements and interactions were verified with a small dataset. Second, in order to determine the sample size of the simulation runs, two different approaches (Cochran, 1963; Yamane, 1967) were used. Four different precision levels (i.e. group sizes) were compared based on the analysis of variance (ANOVA). The analysis results showed that outputs from the four different numbers of simulation runs had similar means and variances. So, a ±10% precision level was chosen to obtain the size of the simulation runs (i.e. 100 simulation runs). It also matched with the minimum size of simulation runs proposed by Helbing and Balietti (2012).

Fifth, in Section 6.6, simulation results were presented and discussed for the ambulance scenario and the breakdown scenario. First, the simulation results of the ambulance scenario showed that the communication messages sharing an ambulance's precise geographic information had a positive influence on minimising the effect of an ambulance's presence on the whole system. Second, the simulation results of the breakdown scenario demonstrated that by broadcasting the breakdown warning to an extended range in advance, recipient vehicles had more time to change lanes without slowing down. In both scenarios, the vehicular communications contributed to minimising the effect of the emergency situation on the whole traffic flow. The simulation results also showed that the effect of vehicular communications on the traffic efficiency in emergency situations might vary depending on situational variables (e.g. road geometry, vehicles' interrelation, etc.).

In conclusion, this chapter gave an explanation of the agent modelling and simulation process, which was designed to supplement the static validation of the VEIN application ontology by providing dynamic situations of vehicular communications. In the simulation, realistic road geography is used to model vehicle agents' movements and reactions in emergency situations. After the ambulance's or the broken-down vehicle's route is defined, its route is used to execute a buffer analysis to remove unnecessary parts of the road network and minimise memory usage during the simulations. For vehicle agents' characteristics in emergency situations, the VEIN application ontology was replicated onto vehicle agent's attributes and methods, mainly in the `CommunicationMessage` class. Consequently, the communication message class can be easily replaced with another class to support the physical communication if necessary.

7. General discussion

7.1 Introduction

The previous three chapters described the three main stages of the methodological framework, which was summarised in Section 1.6. In Chapter 4, the target application area of the thesis was extracted from a list of all ITS services, and two scenarios were developed. Chapter 5 outlined the geospatial ontological framework, which aimed to assist vehicles' context awareness and communications at the domain level, task level, and application level. Chapter 6 presented the agent modelling and simulation, which provided a dynamic virtual road environment and different proportions of communicative vehicles, to examine the situations that are described in the scenarios. Each chapter also included the corresponding results, i.e. the spatiotemporal relations and the relative representation for vehicular communications of the scenarios discussed in Chapter 4, the validation of the ontology model by instantiating vehicles in Chapter 5, and the statistical results of the agent simulations in Chapter 6.

This chapter reviews and evaluates the results of these three chapters, and goes on to discuss the research findings in a broader perspective. Specifically, Section 7.2 discusses the relative location representation, emergency rules, and the SPIN-based approach of the VEIN ontology. Section 7.3 examines whether the agent simulation, which is discussed in Chapter 6, is appropriate to examine the effect of the vehicular communications. Section 7.4 describes general traffic situations and high traffic density to discuss pros and cons and extensibility of the research outcomes. Finally, Section 7.5 reflects upon logical resemblances between the scenarios (Chapter 4), the ontology model (Chapter 5), and the simulations (Chapter 6), and discusses the whole methodological approach from a wider perspective.

7.2 Ontology modelling for sharing spatiotemporally dynamic situations

This section discusses the application ontology of the VEIN ontology model, which was described in Chapter 5. The VEIN ontology model includes a domain ontology, a task ontology, and an application ontology, and these ontologies are developed to support the scenarios at different levels. The VEIN application ontology, which contains classes, instances, and properties, as well as SPIN functions and SPARQL query templates, was produced as the application layer of the intelligent vehicle implementation. Consequently, this section mainly examines the application ontology, as it provides the fundamental ontology of the intelligent vehicle implementation. The following sections compares the VEIN application ontology with other ontology models, discusses SPIN-based approach and ontology evaluation of this research.

7.2.1 Comparisons with other ontology models

This section compares the VEIN ontology model with existing top-level ontology (i.e. DOLCE) and other ontological research related to vehicles' interrelations and communications, which are mostly reviewed in Chapter 2.

DOLCE and its four super-classes (i.e. `Endurant`, `Perdurant`, `Quality`, and `Abstract`) were referred in the process of building the VEIN ontology model. An emergency vehicle and neighbour vehicles are seen as instances of the `Vehicle` class, which is categorised into the `Endurant` class of DOLCE. Vehicles' movements and communications are categorised into the `Perdurant` class. The relative location representation and emergency rules are used to trigger neighbour vehicles' reaction in the vicinity of an emergency vehicle. A vehicle's spatiotemporal properties (e.g. remained distance to a junction) belong to the `Quality` category, while the relative location representation and emergency rules are categorised as a measurement basis and mathematical/logical axioms of the `Abstract` class, respectively. DOLCE provides classes and their hierarchy, but the concepts of the VEIN ontology are represented as properties, SPIN functions and templates. Therefore, it is unable to have one-to-one matches between DOLCE classes and VEIN classes/properties/functions/templates. Still, DOLCE provids a good conceptual foundation to build an application ontology supporting a dynamic traffic situation.

In terms of defining semantic interrelation between vehicles, the relative location representation and emergency rules of this research can be seen similar to the concept of the motion relations (`isBehind`, `inFrontOf`, `driveBeside`, and `passBy`), which Hornsby and King (2008) proposed. They also provided SQL queries to compute the motion relations. They also assumed that an underlying sensor network captures positional data about moving objects in real-time. In this environment, each vehicle's driver can extract specific 'motion relation' information by querying and reasoning from the spatiotemporal sensor database that is linked with an ontology of vehicle classes. However, as described in Section 5.4.3, continuous semantics dealing with high frequency data streams and storing/querying streaming data are still challenging (Mladenić et al., 2012; Stuckenschmidt et al., 2010; Valle, 2014). To overcome this limitation and avoid overloads of reasoning, the ontology model of this research uses SPIN functions and templates that can be processed in memory to keep each vehicle's ontology model up to date.

Eigner and his colleagues (2008, 2009) adopted ontological context model compatible with vehicular communications. However, relationships between ontology classes and specific queries were not represented. Michaels et al., (2010) and Barrachina et al. (2012) proposed communication message models for an emergency vehicle and a vehicle involved in an accident, respectively. In the message, absolute coordinates are main source for the location of the message sender. In contrast, the VEIN application ontology adapted relative location representation as main location source, and the request (i.e. emergency rules) of a message sender (i.e. an emergency vehicle, broken car) could be made based on the relative locations of the message sender and potential message receiver (i.e. a neighbour vehicle). One benefit of using relative location representation is that a local event detection can be done based on a simple triple-based information extraction without geometric calculations. Based on the relative location representation and the triple-based information extraction, a request message from the message sender can be set easily, and a network distance (a.k.a road distance) will be applied automatically for the request range.

To sum up, the VEIN application ontology uses the relative location representation and emergency rules. This approach can extract triple-based information simply without geometric calculation, and it can be utilised for traffic situation detection.

7.2.2 SPIN-based ontological messaging model

A communication message has to contain partial information of a traffic situation, which might change as vehicles move, as traffic variables are dynamic in real time. Consequently, up-to-date information about the situation is necessary for vehicles at each point, and each vehicle's spatiotemporal properties are updated by SPIN functions and SPIN templates in the VEIN application ontology.

The combination of SPIN functions and SPIN templates was useful for the instantiation of a vehicle in the VEIN ontology model. In the model dealing with moving vehicles, assertions are mostly related to their locations, which constantly change. If the model were to use data-driven inferences to calculate and compare vehicles' geometries, increases in the number of inferred triples would be observed. Thus, two SPIN templates are presented to avoid overloads of inferences.

The first template obtains assertions of a vehicle's relative location (distance/time) dynamically on demand from two-dimensional coordinates by referencing road elements and junctions in SPIN functions. These SPIN functions act in a similar manner to a scalar-valued function/method of a programming language that accesses data and returns a single value, such as a string, number, or bit value. The first SPIN template can be seen as a registered continuous query, which gets streams of answers from input streams. Anicic et al. (2012) used a registered continuous query to search slow traffic events and automatically modify a speed limit on a certain road section for the central management. However, in this research, it is used as an internal process to keep a vehicle's location-related properties up to date avoiding the burden of reasoning processes.

In the second template, a DSRC message, providing the relative location information of the ambulance or the broken-down vehicle, is used as input in the SPIN template and the SPIN function of a vehicle's ontology model in order to insert, update, and delete the instance of the ambulance or the broken-down vehicle. For a semantic application in the vehicular environment dealing with moving vehicles, it seems that SPIN offers a good alternative solution, that is, a query-driven backward reasoning, to avoid overloads of inferred data and to integrate with other technologies (e.g. DSRC in this case).

We can conclude that the relative location representation a SPIN-based approach can make a vehicle's decision-making process simple and avoid resource-intensive information extraction. The next section discusses the evaluation approach of the VEIN application ontology and introduces additional evaluation approaches that access an ontology by examining different levels of contents.

7.2.3 Ontology evaluation

The thesis demonstrated in Section 5.5 how the interactions and communications amongst vehicles in the scenarios can be implemented by DSRC messages that instantiate vehicles and their spatiotemporal properties/relations based on the VEIN application ontology. This section discusses the approach to evaluate the VEIN application ontology in comparison to other approaches.

An ontology can be evaluated using various approaches, which can be classified into the following four categories (Brank et al., 2005): comparing the ontology with a 'golden standard' (which may itself be an ontology), comparing the ontology with a source of data (e.g. a collection of documents) by checking whether the ontology covers the concepts of a domain (Brewster et al., 2004), using the ontology in an application and evaluating the results (Porzel and Malaka, 2004), and evaluating the ontology by assessing how well the ontology meets a set of predefined criteria, standards, requirements, etc. The aforementioned evaluation methods can be recategorised based on the different content levels of ontology to examine whether particular ontology contents fit a specific domain, as shown in Table 7.1.

Since the VEIN ontology model has been developed for a specific application area, that is, short-range vehicular communications in emergency situations for an ITS environment, the evaluation of the VEIN application ontology has been limited to the application-level contents. Therefore, the application-level contents of the VEIN ontology model were evaluated by the query results (Sections 5.5.2 and 5.5.3) of the given sequential tasks (Section 5.5.1). The instance examples of the given sequential tasks (Figure 5.10 and Figure 5.11) and the return values of the SPIN query templates (Table 5.4 and Table 5.6) of the ontology model were seen as potential applications implementing the two scenarios, so that the approach used here could be classified into the third category (i.e. using the ontology in an application and evaluating the results) above.

Table 7.1 - Different content levels of an ontology for evaluations (Brank et al., 2005)

Level	Evaluation targets
Lexical, vocabulary, or data layer	Concepts, instances, facts, etc.
Hierarchy or taxonomy	Hierarchical *is-a* relations between concepts
Other semantic relations	Non-taxonomic relations besides *is-a* relations
Context or applications level	How the results of the application are affected by the use of the ontology
Syntactic	Syntactic requirements and considerations (e.g. avoiding a loop between definitions)
Structure, architecture, design	Pre-defined design principles or criteria; structural concerns

However, the instantiation of vehicles and their interactions based on specific sequential tasks was undertaken statically and conceptually and not within the context of a real application. Therefore, it might be controversial whether the instantiation of the VEIN application ontology can be considered as outputs of an application. To supplement this weakness, agent simulations are executed to represent vehicle instances in dynamic situations, and the discussion about agent simulations is explored next.

7.3 Suitability of the agent simulation for vehicular communication

The simulation results were summarised and analysed in Section 6.6 to assess how much vehicles' individual communications would affect a whole traffic situation, especially in emergency situations. Even though vehicular communication can be described as one vehicular network, actual communications can be realised only by individual vehicles (and road facilities) that have a communication device to send and receive situational information to each other.

The concept of the VEIN ontology model is replicated in a communication message class for the simulation, and each vehicle's movements and communications are simulated. The following two subsections discuss the vehicular mobility model and vehicles' communication-based interactions applied in the simulation.

7.3.1 Vehicular mobility model

To examine vehicular communication, a road environment and a traffic flow have to be generated first. For the simulation of this research, a mobility model was implemented to model each vehicle's dynamics relating to its location, velocity, and the distance/time to the lead vehicle or a junction. A microscopic vehicular mobility may consist of car-following model, overtaking model, and junction/intersection model (Härri et al., 2007). This section discusses these three components implemented in the simulation of this research.

First, for each vehicle's microscopic mobility, the Intelligent Driver Model (IDM) was adapted as a car following model. Since IDM is a mathematical model, it cannot fully explain drivers' perception and decision-making process, but it is practical to model mobility patterns of a following vehicle to avoid collision with the lead vehicle (Treiber and Kesting, 2013). According to Kerner (1999), there can be three phases of traffic, which are free flow, synchronized flow, and wide moving jams. In the simulation, a low traffic density was applied and each vehicle's preferred speed was set in the initialisation procedure based on the statistics of free-flow speed ranges. Throughout the simulations, vehicles approaching the lead vehicle or a junction slowed down, and synchronised flows with no significant stoppage were observed.

Second, an overtaking model is partly implemented for the breakdown simulation. As described in Section 5.6, this research emphasised the vehicles' longitudinal interrelation, and the ontology model and the simulation was designed in that way. Therefore, multi-lane roads and an overtaking model are not implemented in this simulation even though the breakdown simulation allowed restricted lane changes to avoid the breakdown.

Third, a simple non-signalised junction management model was implemented based on the kinematic deceleration of the IDM and the first-in-first-out (FIFO) principle. The FIFO principle is violated in real traffic, and there are non-strictly FIFO diverging rule models for signalised urban intersections in the domain of traffic flow modelling. However, the focus on this simulation was not analysing the traffic flow at an intersection, but providing a realistic synchronised traffic forming near roundabouts and intersections. In this regard, the FIFO model was appropriate.

To sum up, the car-following model and the simple FIFO model applied to the simulation was appropriate to reproduce synchronised flows, and the overtaking model was not considered because a low traffic density was assumed.

7.3.2 Vehicular communications as triggers of vehicles' proactive and reactive interactions

This section explores the appropriateness of the agent simulation environment in terms of examining the effect of vehicular communications. From this perspective, the simulation results and agent's communication messaging model are reviewed separately.

First, the simulation results showed the cause-and-effect relationship between vehicles' individual communications and traffic flow efficiency in an emergency situation. According to Epstein (2008), a modelling/simulation approach can be applied for various reasons (i.e. to predict or explain a phenomenon, guide data collection, illuminate core dynamics, suggest dynamical analogies, raise new questions, etc.) if its results provide stationary distributions and regularities of interest. The agent simulation delivered statistical distributions and macroscopic regularities, which emerged from individual vehicles' local interactions, over many simulation runs. In the simulation, vehicles' dynamic and complex interactions were simplified in order to focus on the effect of communicative vehicles, which share situational information and behavioural rules in emergency situations. Accordingly, during the simulation, vehicular interactions in predefined emergency situations were captured not only to explain the effect of vehicular communications, but also to illuminate the core dynamics of vehicular communications.

Second, during the simulation of emergency situations, vehicle agents' proactive and reactive interactions occurred based on their communication-based behaviours, which was tightly linked with vehicular communications. For the ambulance scenario, the ambulance stated its priority to the vehicles in its immediate vicinity and the neighbouring vehicles gave way to the ambulance. For the breakdown scenario, the broken-down car warned the vehicles heading towards the scene of its presence, and the recipient vehicles moved over to the right-hand lane. The proactive behaviour of the message sender vehicle and the recipient vehicles' reactive behaviour were both triggered by emergency rules, which were part of their own messaging model. Vehicle agents shared information about their presence (i.e. location and route) and traffic situations by sending/receiving vehicular communication messages. These communication messages were then used to update the vehicles' declarative memory[37], which was stored as instances and properties/relations in their own messaging model. During the updating process of their messaging model, each vehicle updated its relative location and the emergency vehicle's request. Then, each vehicle checked its emergency rules based on its own procedural memory[38] to make sure whether it needed to perform particular types of action. This circular process enabled each vehicle to keep up-to-date with its messaging model and to use its instances and properties as parameters of the emergency rules.

In summary, the vehicles' movements and communication-based interactions were measured as the simulation results. In addition, we have discussed how vehicle agents' communication messaging model could link vehicle agents' behaviours. The next section discusses general traffic situations whether the agent simulation can support the transition towards intelligent communicative vehicles.

[37] Declarative memory stores specific personal experiences and factual information (e.g. vehicle's location and partial route in the communication messaging model replicating the VEIN ontology model).
[38] Procedural memory is for the performance of particular types of action. A vehicle agents' procedural memory logically corresponds to SPARQL functions and query templates of the VEIN ontology model.

7.4 From limited situations to general situations

This research was focused on the limited emergency situations (i.e. the ambulance scenario and the breakdown scenario), that has relatively low traffic volume and an emergency vehicle sends a message to neighbour vehicles (i.e. one-way communication). However, there can be various communication types to support different situations. In high traffic volume, vehicle agents' behaviour can also be different. This section explores different traffic situations to discuss pros and cons and extensibility of the scenario, the ontology model, and the simulation of this research.

7.4.1. Different communication types for different situations

By message delivery schemes, broadcasting types can be divided into five ways: anycast (one to any), broadcast (one to all), multicast (one to a group), unicast (one to one), geocast (one to a specific area). The communication in the ambulance scenario can be seen as geocast, and one-way communication is preferred as a time-critical solution. It is because the ambulance may expect answerbacks from the neighbour vehicles by their instant give-way reactions rather than sending messages back.

However, there are other situations that two-way communication is necessary. Payment-related applications such as toll payment and parking slot reservation contain a financial transaction during the process, and two-way communication (i.e. unicast) is required. For these kind of application, DSRC is not necessary. Radio Frequency IDentification (RFID), Near-Field Communication (NFC) and mobile communication can be applied instead.

In a situation dealing with fast moving vehicles but not time-critical, two-way DSRC can be used. The reservation-based control at an intersection (Dresner and Stone, 2004, 2005) or the cooperative route-allocation to avoid congestion (Desai et al., 2013) can be a good example of two-way V2I communication (i.e. geocast) and two-way V2V communication (i.e. geocast or multicast), respectively. The case of freeway traffic to resolve a stop-and-go wave (Kesting et al., 2008) can be seen as a multi-hop communication (i.e. anycast).

To sum up, this research chose a one-way single-hop V2V communication (i.e. DSRC, geocast), but there can be various communication types suitable for different situations. The communication types categorised by direction of communication (one-way, two-way), forwarding of message (single-hop, multi-hop), and message delivery schemes (anycast, broadcast, multicast, unicast, geocast) (Daraghmi et al., 2013; Kargl et al., 2006).

7.4.2. Vehicle agents' behaviour in heavy traffic

The ontology model and the simulation framework are developed based on two emergency situations on the assumption that there is no heavy traffic. Based on this assumption, intelligent vehicles' BDI model was simplified and their cooperation resolving an emergency situation is modelled and simulated. However, in heavy traffic, even not in an emergency situation, communicative vehicles and infrastructure should elicit their coordination and negotiation to improve road safety and efficiency.

The simple FIFO-based road junction management model cannot be applied in the situation of traffic congestion. A traditional traffic control operation (i.e. traffic lights) or intelligent intersection control based on V2I and V2V communications should be applied to model a realistic traffic flow. If there is a communicative intersection controller, vehicles approaching the intersection can follow its coordination, which is based on real-time analytics of traffic conditions. Vehicles can pass through the intersection smoothly without the danger of collision or becoming entangled at the intersection (Yang et al., 2014). The intelligent intersection controller can also interact with neighbour intersection controller or an emergency vehicle to coordinate a green light wave to allow continuous traffic flow in one main direction (Bonomi et al., 2012).

The breakdown situation can be extended to a lane closure situation, in which traffic merges into a reduced number of lanes (e.g. from three lanes to two lanes). In the breakdown situation with low traffic density, the emergency rule of the ontology model was designed to trigger the early merging phenomena between two lanes (i.e. the lane of the breakdown and the right lane). Consequently, the simulation was designed such that merging traffic on the lane of the breakdown changes lanes as early as possible after the detection of the breakdown, but yields to through traffic on the right lane they wish to enter. However, traffic congestion may occur when dealing with high traffic volume on multiple lanes, and the merging pattern could be more complex. In this situation, many authorities encourage drivers to use the late merge (zipper merge) method to reduce speed differences between lanes and increase safety. So, a zipper merge model needs to be added to the ontology model and the simulation framework, and an event-driven process is also required to notice traffic congestion and activate the zipper merge model to change the vehicles' reactions. Besides, to minimise complex zipper effects on multiple lanes (e.g. two zippers from three lanes to two lanes or three zippers from four lanes to three lanes) in traffic congestion, a coordinate rule/logic can be designed to make vehicles on all lanes equally disadvantaged to reduce queue backups. A roadside beacon broadcasting a coordination rule can be implanted on the signboard of the lane closure. For the coordination, vehicles (or vehicle platoons) can be asked to do different behaviours (keep the lane or change to the right lane) based on which lane they are on and how far from the lane closure.

When modelling complex situations of high volume traffic, the detailed traffic models explaining traffic control and drivers' behaviours have to be implemented into the ontology model and the simulation framework. In general situations, Vehicle agents' BDI model cannot be simplified anymore because vehicles have long-term goals and plans that can conflict with each other. Vehicles' desires and intentions can be shared for a negotiation via communications. Desai et al. (2013) suggested a cooperative route allocation to avoid congestions, and each vehicle communicates with each other to share its initial route and compute individual welfare and overall welfare to decide its revised route at every junction. This kind of situation can be a practical scenario to extend the BDI model of the ontology model and the simulation framework. The concepts (i.e. BDI, action, policy) of SOUPA (Chen et al., 2004, 2005) can also be applied to link the BDI model and a negotiation rule/behaviour.

To sum up, additional coordination/negotiation rules can be made and shared as parts of communication messages among intelligent vehicles and infrastructure to manage/control these complex situations safely and efficiently. If a communication sender (e.g. intersection controller, roadside beacon) has higher authority than communication receiver (e.g. vehicles nearby), its controlling rule triggering different behaviours from vehicles nearby can be seen as a coordination rule. In a situation of large-scale, long-lasting congestion over a wide area, central management authority can intervene directly in the operation of traffic control or via Center-to-Infrastructure communication (i.e. hierarchical coordination). Meanwhile, if communication messages are sent and received between vehicles, controlling rules need to contain conditions to reach agreements on the situation so that the rules can be considered as negotiations between autonomous vehicles. Of course, there can be a situation that traffic flow is affected by the interplay between hierarchical coordination and autonomous negotiation.

7.5 Methodological discussion

The previous three sections discussed the results of the three main stages of the methodological framework. We now turn to the whole methodological approach. To provide an insight into the further development of geo-ontologies and simulations for future scenarios and applications, the methodological approach is reviewed and reflected upon.

The use of a list of ITS user services, which was considered as an entire logical organisation of ITS applications, proved to be an important tool for extracting a specific target application. Though the use of user services and scenarios is common, the conceptual link of a logical organisation and a spatiotemporal organisation as an integral part of the scenario development is an interesting aspect of this research. This research demonstrates that such an approach can help in understanding the technological perspective (i.e. logical organisation) and situational perspective of an application (i.e. spatiotemporal organisation) when developing a futuristic scenario (Figure 7.1). A holistic perspectives of the ITS environment can facilitate the modelling of intelligent vehicles as individual organisations and members of an organisation simultaneously.

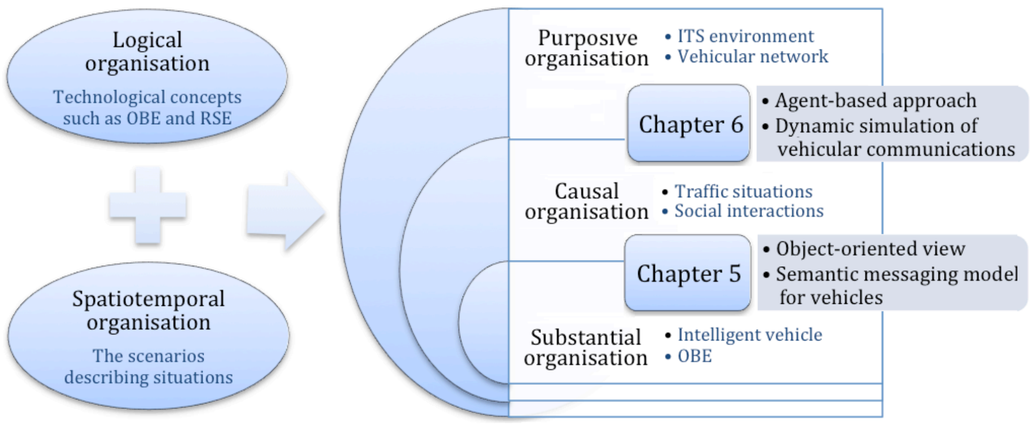


Figure 7.1 - Relations between organisation concepts[39] in the thesis

Since the concepts of the scenarios and spatiotemporal organisations are compatible, the conceptual relations of organisations that are shown in Figure 7.1 can be explained in a simple

[39] According to Krikorian (1935, p.122-124), an organisation can be defined as 'a collection of different *elements* in a set of *relations* forming *a whole*', and it can be classified into five different types depending on the character of elements, relations, and the whole, i.e., logical organisation, spatiotemporal organisation, substantial organisation, causal organisation, and purposive organisation.

way using the scenarios throughout this thesis. Chapter 4 states that the meaning of the term 'scenario' has been limited to represent story-like static snapshots of dynamic situations, which are the two DSRC scenarios. However, in general, a sequence/structure or a simulation can also be considered as a type of scenario from a broader perspective (Alexander, 2004). Therefore, all three main outcomes can be regarded as scenario-based approaches to describe traffic situations in different perspectives (Table 7.2).

Table 7.2 - Scenario-based approaches to describe traffic situations

Type	Description	Contents	Perspective
Scenario	Narrative description of a traffic situation	Vehicles' location, their spatiotemporal relations	Overall snapshot
Ontology	Computational description of vehicle objects, their properties and relations	Properties and relations, SPARQL templates	Object-oriented, sequential
Agent	Computational description of interactions between intelligent vehicle agents in the dynamic processes of traffic flow	Entities, behavioural rules	Dynamic, interactional

From the ITS user services representing logical organisations and two scenarios representing spatiotemporal organisations in Chapter 4, a geospatial ontological framework was developed in Chapter 5, and the ontology model was examined with time sequences of vehicular interactions. Then, in Chapter 6, simulations of vehicular communications were carried out to model dynamic situations and measure the effect of vehicular communications. The ontology model and the agent simulation were developed as a data layer and an application layer that support the vehicular communications to resolve emergency situations, which can be seen as organisational rules in emergency. In the data level, some ontology properties, SPIN functions and templates of the VEIN ontology were used for a vehicle to recognise the situations and then decide whether it has to take action or not, while these properties, query functions and templates are replicated for vehicle agents' behaviours in the simulation.

The VEIN ontology model was designed to support vehicle agents' cooperation, communication, and decision-making in limited emergency situations. However, the ambulance scenario can be used for a variety of emergency vehicles, and the breakdown scenario can be used for road traffic accidents, road maintenance, and road repair. In general, if a situation can be described using spatiotemporal relations between the proactive participants of the situation and some temporal organisational norms can be used to resolve the situation, the ontology model can be extended to support the situation. Temporal organisational norms and their effects on the whole vehicle agent society in the situation can be measured in the agent simulations.

It should be noted that the idea of the modelling and simulation methodologies was borrowed from computer science. In many cases in computer science, a model is described by a formal language to specify particular kinds of systems, or it is embedded in a computer program that simulates the particular behaviours of a system. Since this research has dealt with the messaging model for vehicular communication, these two options are not mutually exclusive; thus, both options were chosen as main parts of the methodology. A geo-ontology model was developed from the scenarios, and then a communication messaging model representing the concept of the ontology model was replicated onto the vehicle agents' behaviours for the simulations.

As a whole, the research has demonstrated that the scenario-based approach is useful as a development scheme for building an ontological framework. It also showed that an agent-based simulation environment is useful to examine the effect of vehicular communications on the traffic flow in a situation of the scenarios. Even though agents' communication-based interactions are simulated in this research, there is an on-going research toward vehicular computing and information-centric networking that proposes a common virtual platform (i.e. inter-networked resources) consisting of vehicles' data storage, sensors, and computing resources (Gerla et al., 2014). In addition, JENA reasoning engine (http://jena.apache.org) and SPIN API (http://topbraid.org/spin/api/) have to be implemented to activate an ontology model in a vehicular computing platform. It can be one further step towards vehicular context-awareness.

7.6 Chapter summary

This chapter discussed the ontology model, the agent simulation, and the methodological approach of this research to support vehicle agents' communication in the road environment to resolve emergency situations. Section 7.2 compared the VEIN ontology with other ontology models and then focused on the relative location representation and emergency rule to simplify vehicle's decision-making process. Benefits of a SPIN-based approach were justified. It also explained how two sequential tasks (Figure 5.10 and Figure 5.11) and query outcomes (i.e. the return values of the SPIN query templates in Table 5.4 and Table 5.6) were used for the evaluation. Section 7.3 discussed the suitability of the agent simulation focused on the vehicular mobility model and vehicles' communication-based behaviour. 7.4 explained how the ontology model and the simulation could be extended to support general traffic situations. Section 7.5

highlighted the conceptual links between the logical and spatiotemporal organisations to consider technological and situational aspects of building a futuristic scenario. The concept of the scenarios (i.e. spatiotemporal organisations), the ontology model, and the agent simulation was discussed from a broader perspective to outline the similarities of the traffic situations that were described in Chapters 4, 5, and 6. It also described how the scenarios and the ontology model could be extended and how the organisational norms of the emergency situations could be measured.

8. Conclusion

The area of intelligent vehicle applications can be charted along two dimensions: the degree of *autonomy* and the degree of *cooperation* (Silberg et al., 2012). Recently, there has been fast-growing attention and technical progress for the development of autonomous cars[40]. The automotive industry has mainly focused on the design and development of sensor techniques (e.g. radar, lidar, GPS, computer vision, etc.) for an autonomous car to increase the degree of autonomy and ultimately, to navigate without human input. Even though vehicular communication technologies for vehicular cooperation receive less attention than sensor technologies for autonomous cars, the high degree of vehicle autonomy creates the need also for a higher degree of vehicular cooperation in many ITS applications (Silberg et al., 2012). In addition, physical communication technologies (e.g. DSRC) are becoming mature and are becoming the basic cornerstones for V2V and V2I environments. For the dimension of vehicular cooperation, this research has explored the development of appropriate communication contents among vehicles on top of physical communication technologies.

Considering the importance of a vehicular communication network and its contents, this research has attempted to build a vehicular version of a Semantic Web by developing a geospatial ontology model that could be used as a vehicle's context model as well as communication contents. This research also examined vehicles' communications using both static and dynamic instantiations of vehicles and their interactions. For the dynamic instantiations, a virtual road environment and vehicle agents were simulated focusing on the short-range vehicular communications of specific traffic situations (i.e. local emergency situations). The following sections revisit the research questions and then outline the research outcomes, the research contributions, the research limitations, and suggestions for future work.

[40] It is also known as a robotic car, a driverless car, or a self-driving car. However, throughout this thesis, the term 'autonomous' has been used to refer specifically to vehicle agents' flexible autonomous behaviours based on their social abilities (i.e. cooperation, coordination, and negotiation) to resolve emergency situations by providing information/warning to assist drivers.

8.1 Reflecting on the research questions

The main aim of this research was 'to explore how *geosemantic* data/information can be used to make a transport system more *intelligent*' based on the vehicular communication in an ITS environment. This thesis was intended to achieve this research aim by answering the research questions, which were described in Section 1.5. This section summarises and discusses how these research questions were addressed throughout the research undertaken.

- *Which area of ITS applications will benefit most from intelligent vehicles and their short-range communications? Which traffic situations should be modelled to generate this benefit?*

This research question was formulated to find time-sensitive dynamic traffic situations, in which geographic information has a significant role to resolve the situations. In order to assist vehicles in these kinds of situations, spatiotemporal information representing moving vehicles' location, route, and interrelation had to be the main parts of the description of traffic situations.

To answer this research question, the two scenarios that represented vehicular communications in emergency traffic situations were developed, through following a two-step process. In the first step, from a well-defined list of the ITS user services (Section 4.2.1), two ITS service bundles (Sections 4.2.2 and 4.2.3) were highlighted to represent immediate vehicular communications in emergency traffic situations. In the second step, the target application area (Section 4.2.4) was specified on the basis of whether real-time communication (sending and receiving real-time spatiotemporal information) plays an indispensable role to cause vehicles' instantaneous interactions. Finally, two scenarios (i.e. the ambulance scenario in Section 4.3.1 and the vehicle breakdown scenario in Section 4.3.2) were developed to describe vehicular communication in local traffic situations representing the target application. The scenarios represented proactive vehicles (i.e. an ambulance and a broken-down vehicle) and reactive neighbouring vehicles in the vicinity of the emergency situations.

- *What are the core contents of the vehicular communications in the traffic situations?*

The second research question was asked to specify what kind of information could comprise communication messages in the specific traffic situations, which were already defined from the first research question.

In Section 4.4, dynamic inter-vehicle relations and relative location representation were emphasised as potential parts of communication contents. Subsequently, In Chapter 5, a geospatial ontological framework (i.e. VEIN ontology) was proposed as a network-centric communication contents model to describe vehicles' dynamic locations and relations. In the ontological messaging model, relative location information of an emergency vehicle was the one part of the core contents and an emergency rule that describes the emergency vehicle's request was the other. With relative location representation, two vehicles' relative distance/time could be acquired by simple comparison between two scalar values. It enables to make the emergency rule without geometric calculations. The emergency rules were defined as query templates, so vehicles could cooperate with the emergency vehicle by performing an action based on a query result.

- *How can the contents of the vehicular communications be linked effectively with intelligent vehicles' actions?*

An intelligent vehicle agent takes action based on its internal states and external environment. A vehicle's internal states can be represented by its beliefs (i.e. knowledge-base), desires (i.e. goals), and intentions (i.e. plans for actions). Furthermore, for a vehicle in a traffic flow, its neighbouring vehicles can be considered as parts of its external environment. Thus, descriptions of vehicles' presences and routes were the main part of the vehicular communication contents to share/resolve traffic situations. However, even though vehicles could send and receive situational information that was closely linked to vehicles' internal states via vehicular communications, communication messages themselves were not the main body of action. There was a need for a linkage between the communication message model (i.e. VEIN application ontology) and the vehicle agents' properties and methods that trigger their reactions.

The development of the agent-based simulation platform, which modelled vehicles' movements, communications, and interactions in emergency situations, was described in Sections 6.2 to 6.4, as the process to link the vehicular communication messages (i.e. a replication of the VEIN ontology model) and the agent's properties and methods. As this research deals only with emergency traffic situations, a vehicle agent's beliefs, desires, and intentions (BDI) could be easily simplified as a vehicle agent's properties and methods to take action in a way that resolves an emergency situation (Section 6.2). A virtual road environment was built to simulate vehicle agents' movements and communications (Section 6.3). For virtual communications to trigger vehicle agents' reactions, a Java class (i.e. `CommunicationMessage`) was developed on top of the Repast Simphony platform as a replication class of the VEIN application ontology (Section 6.4).

- *If communicative vehicles have an influence over the whole system, how can the impact be measured?*

  This research focused on the communications between an emergency vehicle and its neighbouring vehicles (i.e. an emergency vehicle's proactive requests/warnings and its neighbour vehicles' reactions). To model vehicles' microscopic mobility, a car-following model and a FIFO-based junction management model are implemented in the agent simulation framework (Section 6.2). During the simulation, movements and communication between an emergency vehicle and the neighbouring vehicles are modelled, and the number of vehicles that are not disturbed by an emergency vehicle is counted (Section 6.4). Even though direct communications among neighbouring vehicles were not considered, the simulation results showed increasing trends of traffic efficiency with the increase of DSRC-equipped vehicles and their communications (Section 6.6).

We have revisited the research questions and seen how the research framework was built to answer the research questions. Even though this research focused on local emergency situations, each component of this framework can be easily extended and modified for the design and development of intelligent vehicle agents in other traffic situations. The next section describes the research outcomes (i.e. the scenarios, the ontology model, and the agent simulation), which have been developed and discussed throughout this thesis.

8.2 Research outcomes: scenarios, the ontology, and the agent simulation

After the literature review of the ontological and agent-based approaches in the intersection area of the ITS-related domains, a three-level (i.e. description level, data level, and application level) research procedure was undertaken for the implementation of intelligent vehicles throughout Chapters 4, 5, and 6.

The main outcomes of this research consist of three components: futuristic scenarios, a geospatial geosemantic messaging model, and an agent simulation environment. These three components represent a description level (i.e. two scenarios in local emergency situations), a data level (i.e. a geospatial ontological framework to support vehicular communication messages), and an application level (i.e. an agent simulation to model vehicle agents and their interactions) for the implementation of intelligent vehicles. This section outlines these main outcomes and examines their characteristics.

- *Scenarios: an ambulance case and a broken-down car case*

  This thesis provided two specific scenarios (i.e. the ambulance scenario and the breakdown scenario), which were described in Chapter 4. The two scenarios illustrated two different traffic situations: a spatially dynamic influence based on an ambulance's movements and a spatially static influence because of a broken-down vehicle's presence, in which vehicular communications might have a positive effect on resolving the situations.

  Along with the scenarios, this thesis provided two additional ancillary outcomes. First, based on a logical organisational concept, the target application area was extracted to specify the situations, which focused on intelligent vehicles' proactive and reactive characteristics based on vehicular communications. This process for extracting the target application area presented a general process to extract an application area from the whole ITS system. Second, vehicles' locations in the scenarios are represented based on scalar value properties and binary relations with local road segments and junctions rather than their coordinates, focusing on the vehicles' longitudinal interrelations. By simplifying the geosemantic descriptions for vehicles' interrelations and the decision-making processes, the relative descriptions (i.e. vehicles' spatiotemporal relations and relative location representation) set up a base for the ontological modelling of vehicles' communication messages.

- *Geospatial ontological framework: the VEIN ontology*

  To support intelligent vehicles' movements and communications in an ITS environment, the VEIN ontology was developed as a hierarchical ontological framework, which was composed of a domain ontology, a task ontology, and an application ontology. Based on the VEIN domain ontology that provided the four main classes (i.e. vehicles, infrastructure, travellers, and centres) of the ITS architecture, the VEIN task ontology and the VEIN application ontology were developed.

As a network-centric messaging model of AI (i.e. intelligent vehicles), the VEIN application ontology was designed to store declarative (i.e. bottom-up) statements and procedural (i.e. top-down) statements. The declarative statements were used to describe vehicles' relative locations in a specific time, whist the procedural statements defined emergency rules to support vehicles' autonomous interactions to resolve emergency situations. These statements were all written in the triple patterns so that the VEIN application ontology could be seen as a single ontological framework.

- *Agent simulation to model vehicles' movements and communications*

To supplement the static validation of the VEIN application ontology, the agent simulation environment was developed to provide a dynamic virtual road environment, in which vehicle agents were moving and interacting with each other. For the road geography of the two scenarios, two different road datasets were extracted from the ITN layer and then stored as shapefiles. To represent a vehicle agent's internal states in emergency situations, the vehicle agent's beliefs, desires, and intentions (BDI) architecture was simplified onto the vehicle's attributes and methods. The agent simulation environment was implemented using Repast Simphony, and simulation outputs were collected in every 'tick' during the simulations.

A simulation normally includes a proper validation concerning whether the simulation is a good model of the real system. Since this research deals with a futuristic situation, the verification of the emergency rules in a test set was used as the predictive validation of the futuristic situation. The main simulations were executed with different proportions of OBE-implanted vehicles for each scenario, and the result of the regression analysis showed that the effect of vehicular communication in an emergency situation increases linearly when there are more OBE-implanted vehicles in the situation.

In summary, as this research deals with vehicular communication network on a road network, it provides a unique theme for the modelling and simulation of vehicular interrelations. The next section describes the research contributions to the domain of intelligent transport systems.

8.3 Research contributions

This research uses triples of ontology to describe vehicles' spatiotemporal interactions in order to emphasise the network-centric characteristics of a vehicular communication network on a road network. The novel contributions of this research can be described in four aspects: 1) articulating the need of relative location representation and emergency rules as core contents of the communication message to trigger neighbour vehicles' reaction effectively; 2) proposing a SPIN-based ontological messaging model that provides a query-driven backward reasoning to avoid overload of inferred data; and 3) presenting a simulation framework that can evaluate the communications in the emergency situations.

First, this research proposed an ontological approach, which has been used to support the developments of the vehicles' context model and communication messaging model. For a vehicle in a traffic situation, its VEIN ontology model represented its relative location and spatiotemporal relations with other vehicles and/or road elements. With the network-centric descriptions of the ontology model, partial information (i.e. partial ontology) of an individual vehicle could be easily extracted from its whole context model and then merged with another vehicle's partial ontology. By sending and receiving its ontology-based messages (e.g. an emergency vehicle's relative location and the emergency rule), the recipient vehicles could understand the surrounding situation, and then take action to resolve and/or avoid the situation.

Second, a SPIN-based ontological messaging model (i.e. VEIN ontology) was developed to apply the relative location representation and emergency rules in a practical way. In the VEIN application ontology, the relative location of a vehicle was represented by its properties (e.g. remained distance to a node) and relations (e.g. which road it is located on, which node it is heading to). A vehicle's relative location could be easily translated into a distance/time towards a junction or another vehicle so that the vehicle's decision process to resolve a situation could be simplified. In the ontology model, several SPIN functions and query templates were also developed to assist vehicles/drivers by dealing with emergency rules (e.g. giving way or changing lanes). To avoid overloads of reasoning, but still keep the benefits of ontology, this research adopted SPIN functions and templates. Using SPIN functions and templates, it is possible to represent context in programming code level in computer memory, which is a benefit of the object-based context model, and detect emergency situations (i.e. event detection), which is a benefit of the logic-based context model.

Third, this research has validated vehicular communications in a dynamic traffic environment by using an agent-based simulation, which supplemented the static validation of the VEIN application ontology. To replicate the concept of the ontological context/messaging model, a communication message class was developed. Two emergency traffic situations were simulated in order to examine the advantage of sharing situational information via the communication technology on the road. Even though the scope of this research was limited to the predefined emergency situations, the agent simulation environment could be extended for modelling other traffic agents (e.g. traffic signal controller). An agent simulation framework has also provided that could measure the effect of vehicular communications on the traffic efficiency in specific situations. With the agent simulation framework, various situational environments (e.g. different communication ranges, traffic density, emergency rules) can be generated and compared to each other or expected theoretical results.

Yet, as this research has focused on very specific emergency traffic situations, there are research limitations. The next section discusses the research limitations from two different perspectives: research outcomes and research method.

8.4 Research limitations

The previous two sections (Sections 8.2 and 8.3) outlined the main research outcomes and research contributions. Even though this thesis has addressed the research aim, there are limitations that need to be considered. This section describes the limitations of the three main outcomes (i.e. the scenarios, the ontology model, and the agent simulation) as well as the methodological limitations.

- *Scenarios: locality of each scenario and cooperation-focused description*

  The scenarios were focused on building a vehicular communication message model to support vehicular cooperation in specific emergency situations, in which vehicles share geospatial information to resolve a shared traffic situation. Therefore, the scenarios were built on top of the subservices of the ITS user services, and reflected only parts of reality. In a real traffic situation, the locality and independency of each scenario cannot be assumed any more. There would be cases where two or more events might happen in proximity and a vehicle on a road may be in the impact zone of several events, but these multiple traffic situations have not been discussed. When dealing with several concurrent traffic situations, vehicular coordination and negotiation have to be considered in order to prioritise and allocate resources to resolve several situations in sequence.

- *The VEIN ontology: restricted availability for emergency situations that contain point events on a linear network*

The VEIN ontology was developed as a communication message model on the conceptual basis of an *ad hoc* vehicular network (i.e. DSRC) and ontology-based context modelling. As a network-centric message model on top of restricted scenarios, vehicles' locations were described with relative distance and time from/to a road element. Even though relative geographic representation of the VEIN ontology enabled SPARQL queries to be made without spatial indices, it is limited to point features/events on a linear road network. The linear road network model (i.e. one-dimensional linear space) provided a simplified and abstracted description of the real world road network, but it was focused on describing vehicles' longitudinal interrelation, rather than vehicles' lateral interrelation. Consequently, two-dimensional or three-dimensional descriptions, such as roadway width and multi-lane properties, were not considered.

- *The agent simulation: strict adherence to the predefined emergency rules without learning and updating vehicles' behaviours*

The agent simulations focused on examining the effect of the vehicular communication to share vehicles' geospatial information in emergency situations, so vehicle agents' adaptation or learning was beyond the scope of this research. Consequently, despite the fact that vehicles were modelled in an agent-based way, vehicles were quite limited to 'learn and update their behaviours' from a decision-making perspective (Johnston et al., 2013). Drivers' different behaviours could also affect the traffic flow, but were not considered since this research deals with emergency situations that drivers needed to follow emergency rules, which were compulsory for their own safety.

Apart from limitations of the research outcomes, there is another limitation caused by the methodological framework. The research questions were developed like a parallel circuit rather than a series circuit so that each research outcome can be treated as an individual solution of each research question. The VEIN application ontology was developed as a communication messaging model, and the agent simulation was just designed to demonstrate the effects of vehicular communications by linking the communication message to the vehicle agents' behaviours. Even though the communication message class for the simulation was built based on the concept of the VEIN application model by replicating its properties and functions in a Java class, the VEIN application ontology was not physically coupled with the simulation. Therefore, the agent simulation has fundamental limitations and cannot be treated as an independent solution for providing dynamic validation of the VEIN application ontology.

- *Restricted simulation environment: limited to vehicle-to-vehicle communications and static road-related information*

  During the simulation, it was assumed that each vehicle had its own storage for its ontology model not only to keep the whole static road network, but also to update vehicles' presence via OBE-to-OBE communications. Vehicle-to-infrastructure communication (i.e. OBE-to-RSE communication) and vehicle-to-centre communication, which could expand the ontology model for vehicle agents to receive/update dynamic road-related information in realistic traffic situations, were not considered in the simulation. Consequently, ontology storage options for the aforementioned communication types (i.e. road-segment based partitioning for RSE, traffic analysis zone based partitioning for traffic centre) were not considered in the simulation.

Based on the aforementioned research limitations, the next section will recommend directions for future research.

8.5 Future work

The previous section (Section 8.5) provided detailed descriptions of the research limitations, and they can be considered as guidelines to refine the research outcomes and research method. This section outlines recommendations for future research to improve the VEIN ontology and the agent simulation, which can be regarded as the two main columns that support the research framework. Providing more scenarios can extend the VEIN application ontology, and the implementation of vehicle agents can be extended by providing an Application Programming

Interface (API) that reads/writes the VEIN ontology model as well as providing a more sophisticated decision-making framework. The simulation framework can be improved by providing more detailed vehicle mobility models and realistic traffic volumes.

- *Providing more scenarios to extend the VEIN application ontology*

  The VEIN domain ontology and the VEIN task ontology were developed by analysing transport features/elements and vehicular communication devices/elements, respectively. Then, the VEIN application ontology was developed as a limited set of subclasses, properties, and relations as well as SPIN rules and SPARQL templates. The same procedure can be done with other scenarios in order to extend the VEIN application ontology. Additional scenarios can describe vehicle-to-infrastructure communication, other vehicular sensors (e.g. LIDAR, ultrasonic, optic sensors, etc.), and concurrent situations in which vehicles need to coordinate and negotiate.

- *Providing an API to access the VEIN ontology*

  Since the vehicular communications were simulated in a virtual road environment, the VEIN ontology model was logically mapped onto vehicle agents' properties (i.e. internal states) and methods (i.e. behaviours). It was enough to examine whether the ontology model could be regarded as having reliable communication contents to resolve the emergency situations. However, to become an independent messaging model and to be integrated with the simulation, the VEIN ontology model needs to provide an API for accessing its own ontology storage to get/set asserted triples, SPIN functions, and SPARQL templates. In addition, building an ontology API would be the first important step for the physical implementation of intelligent vehicles.

- *Developing advanced decision-making process for intelligent vehicles*

  The VEIN ontology contains SPARQL templates, which represent vehicle agents' current decision-making process. These templates only check if a vehicle meets a predefined situation by measuring the vehicle's relative distance/time from an emergency location/zone. This kind of predefined routines can be powerful when dealing with a simple situation, but an advanced decision-making process for vehicles is essential to deal with various complex situations. With the integration of vehicles' predefined cooperation rules (i.e. emergency rules) and additional coordination and negotiation rules, a stage-by-stage decision-making framework (e.g. situation - options – choose – act – evaluate) needs to be developed.

- *Replicating more realistic traffic situation for the simulation*

  For the microscopic vehicle mobility, the simulation framework only provides a car-following model (IDM) and an unsignalised FIFO-based junction/intersection management. Multi-lane roads, vehicles' overtaking, and traffic controls (e.g. traffic lights at junctions/intersections) can be implemented to reproduce realistic vehicles' movements. For realistic traffic volume at a given moment in a particular area, it is also required to use additional surveys and statistics of real traces of vehicles in real daily life urban traffic over commuting time, lunchtime, weekdays, weekends.

8.6 Concluding remarks

The advancement of Information and Communication Technologies (ICT) has led to the transition of transport systems towards Intelligent Transport Systems (ITS). This research chose emergency situations in an ITS environment as the target application area to illustrate the actuality of a geospatial ontology that could support communications of vehicles to resolve the situations locally and autonomously. The approach outlined in this thesis can be applied to build other ontological models that accentuate spatiotemporal relations. For example, based on spatiotemporal relations among road users and road facilities, a flexible public transportation routing/scheduling model, a speed enforcement model, a traffic signal enforcement model, or a road-geometry warning model can be developed as an application ontology. However, since this research has adopted the agent simulation instead of the physical implementation of multi vehicle agents, several ontological issues (i.e. speed issues, inference issues, storing issues, etc.) need to be considered when contemplating a physical implementation of an ontology model and/or a multi-agent system.

Meanwhile, the ontology-based messaging model demonstrates the potential of vehicle agents that communicate with each other in emergency situations, which can be categorised as special cases. Therefore, to support general cases in various situations in ITS and other domains, the context model for the vehicle agents (and the road infrastructure agents) has to be extended to represent their beliefs, desires, and intentions (BDI). In addition, for the physical implementation of autonomous vehicle agents, it has to be interconnected and compatible with other sensors (i.e. LIDAR, ultrasonic, and optic sensors) and techniques (i.e. information extraction, machine-learning, etc.) to recognise the traffic and road environment.

In conclusion, in order to support communication and sensor technologies, it is necessary to explore a new way for intelligent agents to capture, store, and share geographic information and knowledge on a distributed network without a central geodatabase. In the way of this exploration, the current concept and functionality of geographic information, which is based on the geovisualisation-oriented and geodatabase-oriented framework, would be extended both for human and intelligent agents.

References

Abdou, M., Hamill, L., Gilbert, N., 2012. Designing and Building an Agent-Based Model, in: Agent-Based Models of Geographical Systems. Springer, pp. 141–165.

Agarwal, P., 2005. Ontological considerations in GIScience. International Journal of Geographical Information Science 19, 501–536. doi:10.1080/13658810500032321

Alexander, I., 2004. Introduction: Scenarios in System Development, in: Alexander, I., Maiden, N. (Eds.), Scenarios, Stories, Use Cases : Through the Systems Development Life-Cycle. John Wiley & Sons.

Allan, R.J., 2010. Survey of Agent Based Modelling and Simulation Tools (Technical Report No. DL-TR-2010-007). Science and Technology Facilities Council, UK.

Al-Sultan, S., Al-Doori, M.M., Al-Bayatti, A.H., Zedan, H., 2014. A comprehensive survey on vehicular Ad Hoc network. Journal of Network and Computer Applications 37, 380–392. doi:10.1016/j.jnca.2013.02.036

An, L., Linderman, M., Qi, J., Shortridge, A., Liu, J., 2005. Exploring complexity in a human–environment system: an agent-based spatial model for multidisciplinary and multiscale integration. Annals of the Association of American Geographers 95, 54–79.

Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N., 2012. Stream reasoning and complex event processing in ETALIS. Semantic Web 3, 397–407. doi:10.3233/SW-2011-0053

Apple Inc., 2011. The Objective-C Programming Language.

Architecture Development Team, 2012. National ITS Architecture Physical Architecture. Research and Innovative Technology Administration (RITA), US Department of Transportation (US DOT), Washington, DC, USA.

Architecture Development Team, 2007. National ITS Architecture Documents: EXECUTIVE SUMMARY. Research and Innovative Technology Administration (RITA), US Department of Transportation (US DOT), Washington, DC, USA.

Baader, F., Horrocks, I., Sattler, U., 2009. Description Logics, in: Handbook on Ontologies. Springer, pp. 21–43.

Baldauf, M., Dustdar, S., Rosenberg, F., 2007. A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2, 263–277.

Batty, M., 2001. Polynucleated Urban Landscapes. Urban Stud 38, 635–655. doi:10.1080/00420980120035268

Batty, M., Desyllas, J., Duxbury, E., 2003. Safety in Numbers? Modelling Crowds and Designing Control for the Notting Hill Carnival. Urban Stud 40, 1573–1590. doi:10.1080/0042098032000094432

Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D., 2011. SUMO–simulation of urban mobility: an overview, in: Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation. ThinkMind.

Benenson, I., Omer, I., Hatna, E., 2002. Entity-based modeling of urban residential dynamics: the case of Yaffo, Tel Aviv. Environment and Planning B 29, 491–512.

Bergman, M.K., 2009. Advantages and Myths of RDF. Structured Dynamics LLC.

Bernaras, A., Laresgoiti, I., Corera, J., 1996. Building and Reusing Ontologies for Electrical Network Applications', in: 12th European Conference on Artificial Intelligence. Budapest, Hungary, pp. 298–302.

Berners-Lee, T., Hendler, J., Lassila, O., 2001. The semantic web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American.

Bonabeau, E., 2002. Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences 99, 7280–7287. doi:10.1073/pnas.082080899

Bonomi, F., Milito, R., Zhu, J., Addepalli, S., 2012. Fog Computing and Its Role in the Internet of Things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12. ACM, New York, NY, USA, pp. 13–16. doi:10.1145/2342509.2342513

Booch, G., 1994. Object-oriented analysis and design with applications. Addison-Wesley.

Borst, W.N., 1997. Construction of engineering ontologies. Centre of Telematica and Information Technology, University of Tweenty: Enschede, The Netherlands.

Brank, J., Grobelnik, M., Mladenić, D., 2005. A survey of ontology evaluation techniques. Presented at the Conference on Data Mining and Data Warehouses (SIKDD 2005) at 7th International Multi-conference on Information Society (IS 2005), Ljubljana, Slovenia.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J., 2004. Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems 8, 203–236. doi:10.1023/B:AGNT.0000018806.20944.ef

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J., 2001. A knowledge level software engineering methodology for agent oriented programming, in: Proceedings of the Fifth International Conference on Autonomous Agents, AGENTS '01. ACM, New York, NY, USA, pp. 648–655. doi:10.1145/375735.376477

Brewster, C., Alani, H., Dasmahapatra, S., Wilks, Y., 2004. Data driven ontology evaluation. Presented at the International Conference on Language Resources and Evaluation (LREC 2004), Lisbon, Portugal.

Brickley, D., Guha, R.V., 2004. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation.

Brown, D.G., Page, S.E., Riolo, R., Rand, W., 2004. Agent-based and analytical modeling to evaluate the effectiveness of greenbelts. Environmental Modelling & Software 19, 1097–1109. doi:10.1016/j.envsoft.2003.11.012

Burmeister, B., Haddadi, A., Matylis, G., 1997. Application of multi-agent systems in traffic and transportation. Software Engineering. IEE Proceedings 144, 51–60. doi:10.1049/ip-sen:19971023

Cardoso, J., 2006. Developing An Owl Ontology For e-Tourism, in: Cardoso, J., Sheth, A.P. (Eds.), Semantic Web Services, Processes and Applications, Semantic Web and Beyond. Springer US, pp. 247–282.

CCL, 2011. NetLogo 4.1.3 User Manual. Center for Connected Learning and Computer-based Modelling (CCL).

Chen, H., Finin, T., Joshi, A., 2005. The SOUPA Ontology for Pervasive Computing, in: Tamma, V., Cranefield, S., Finin, T., Willmott, S., Walliser, M., Brantschen, S., Calisti, M., Hempfling, T. (Eds.), Ontologies for Agents: Theory and Experiences, Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser Basel, pp. 233–258.

Chen, H., Perich, F., Finin, T., Joshi, A., 2004. SOUPA: standard ontology for ubiquitous and pervasive applications, in: The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 2004). pp. 258–267. doi:10.1109/MOBIQ.2004.1331732

Choffnes, D.R., Bustamante, F.E., 2005. An integrated mobility and traffic model for vehicular wireless networks, in: Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks. ACM, pp. 69–78.

Cochran, W.G., 1963. Sampling techniques. Wiley.

Computer Desktop Encyclopedia, 2011. OSI Model. The Free Dictionary.

Connolly, D., 2006. Naming and Addressing: URIs, URLs, ... http://www.w3.org/addressing/.

Conte, R., Gilbert, N., Sichman, J., 1998. MAS and Social Simulation: A Suitable Commitment, in: Sichman, J., Conte, R., Gilbert, N. (Eds.), Multi-Agent Systems and Agent-Based Simulation, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 1–9.

Cook, D.J., Augusto, J.C., Jakkula, V.R., 2009. Ambient intelligence: Technologies, applications, and opportunities. Pervasive and Mobile Computing 5, 277–298. doi:10.1016/j.pmcj.2009.04.001

Cook, D.J., Youngblood, M., Heierman, E.O., Gopalratnam, K., Rao, S., Litvin, A., Khawaja, F., 2003. MavHome: an agent-based smart home, in: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003). IEEE, pp. 521–524. doi:10.1109/PERCOM.2003.1192783

Crooks, A.T., 2010a. Using Geo-spatial Agent-Based Models for Studying Cities (No. CASA Working Paper 160), CASA Working Papers Series. Centre for Advanced Spatial Analysis, University College London.

Crooks, A.T., 2010b. Constructing and implementing an agent-based model of residential segregation through vector GIS. International Journal of Geographical Information Science 24, 661–675. doi:10.1080/13658810903569572

Crooks, A.T., 2007. The Repast Simulation/Modelling System for Geospatial Simulation (No. CASA Working Paper 123), CASA Working Papers Series. Centre for Advanced Spatial Analysis, University College London.

Cycorp, 2002. OpenCyc Selected Vocabulary and Upper Ontology [WWW Document]. OpenCyc Documentation. URL http://www.cyc.com/cycdoc/vocab/upperont-diagram.html (accessed 5.25.12).

Daraghmi, Y.-A., Stojmenovic, I., Yi, C.-W., 2013. A taxonomy of Data Communication Protocols for Vehicular Ad Hoc Networks, in: Basagni, S., Conti, M., Giordano, S., Stojmenovic, I. (Eds.), Mobile Ad Hoc Networking: The Cutting Edge Directions. John Wiley & Sons.

Department for Transport, 2013. Transport statistics Great Britain 2013. Department for Transport.

Department for Transport, 2011a. Vehicle speeds on built up roads by speed limit and vehicle type in Great Britain, annual from 2006 (No. SPE0102). Department for Transport, Great Britain.

Department for Transport, 2011b. Vehicle speeds on non built up roads by road type and vehicle type in Great Britain, annual from 2006 (No. SPE0101). Department for Transport, Great Britain.

Department for Transport, 2011c. Vehicle speeds on built up roads in Great Britain, annual from 2005 (No. SPE0104). Department for Transport, Great Britain.

Department for Transport, 2011d. Vehicle speeds on non-built-up roads in Great Britain, annual from 2005 (No. SPE0103). Department for Transport, Great Britain.

Department for Transport, 2006. Setting local speed limits, DfT Circular 1/06 (No. DfT Circular 1/06). Department for Transport, Great Britain.

Department for Transport, 2004. The Future of Transport: a network for 2030. Stationery Office.

Desai, P., Loke, S.W., Desai, A., Singh, J., 2013. CARAVAN: Congestion avoidance and route allocation using virtual agent negotiation. IEEE Transactions on Intelligent Transportation Systems 14, 1197–1207.

Dey, A.K., 2001. Understanding and Using Context. Personal and Ubiquitous Computing 5, 4–7. doi:10.1007/s007790170019

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271. doi:10.1007/BF01386390

Dresner, K., Stone, P., 2005. Multiagent traffic management: an improved intersection control mechanism, in: Proceedings of the Fourth International Joint Conference on

Autonomous Agents and Multiagent Systems, AAMAS ’05. ACM New York, NY, USA, Utrecht, The Netherlands, pp. 471–477. doi:10.1145/1082473.1082545

Dresner, K., Stone, P., 2004. Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism, in: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS ’04. IEEE Computer Society Washington, DC, USA, New York, USA, pp. 530–537. doi:10.1109/AAMAS.2004.190

Dressler, F., Sommer, C., 2015. Simulation of Vehicular Networks.

Driving Standards Agency, 2007. The Official Highway Code. The Department for Transport.

Dulmage, J., Tsai, M., Fitz, M., Daneshrad, B., 2006. COTS-based DSRC testbed for rapid algorithm development, implementation, and test, in: Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization, WiNTECH ’06. ACM, New York, NY, USA, pp. 113–114. doi:10.1145/1160987.1161017

Ebers, S., Hellbück, H., Pfisterer, D., Fischer, S., 2013. Collaboration between VANET applications based on open standards, in: 2013 IEEE Vehicular Networking Conference. Presented at the 2013 IEEE Vehicular Networking Conference, pp. 174–177. doi:10.1109/VNC.2013.6737606

ECC, 2008. ECC Decision of 14 March 2008 on the harmonised use of the 5875-5925 MHz frequency band for Intelligent Transport Systems (ITS) (No. ECC/DEC/(08)01). Electronic Communications Committee (ECC), European Conference of Postal and Telecommunications Administrations (CEPT).

Eichler, S., Ostermaier, B., Schroth, C., Kosch, T., 2005. Simulation of car-to-car messaging: analyzing the impact on road traffic, in: 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. Presented at the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005, IEEE, pp. 507–510. doi:10.1109/MASCOTS.2005.64

Eigner, R., Lutz, G., 2008. Collision Avoidance in VANETs - An Application for Ontological Context Models, in: Sixth Annual IEEE International Conference on Pervasive Computing and Communications. pp. 412–416. doi:10.1109/PERCOM.2008.42

Eigner, R., Mair, C., 2009. Using Context Ontologies for Addressing and Routing in Mobile Ad Hoc Networks, in: Eighth International Conference on Networks. pp. 415–420. doi:10.1109/ICN.2009.72

EMSA, 2009. EMSA’s New Sirens Shake Drivers to Attention [WWW Document]. URL https://www.emsaonline.com/mediacenter/articles/00000622.HTML (accessed 7.28.14).

Epstein, J.M., 2008. Why Model? Journal of Artificial Societies and Social Simulation 11, 12.

Epstein, J.M., Axtell, R., 1996. Growing artificial societies: social science from the bottom up. Brookings Institution Press.

ESRI UK, 2006. Using OS MasterMap® Integrated Transport Network (ITN) Layer with ArcGIS (ESRI (UK) White Paper).

ETSI, 2008. ETSI EN 302 571 V1.1.1, Intelligent Transport Systems (ITS); Radiocommunications equipment operating in the 5 855 MHz to 5 925 MHz frequency band; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive (No. DEN/ERM-TG37-007). European Telecommunications Standards Institute (ETSI).

ETSI, 2006. ETSI TR 102 492-2 V1.1.1, Electromagnetic compatibility and Radio spectrum Matters (ERM); Intelligent Transport Systems (ITS); Part 2: Technical characteristics for pan European harmonized communications equipment operating in the 5 GHz frequency range intended for road safety and traffic management, and for non-safety related ITS applications; System Reference Document (No. DTR/ERM-RM-036-2). European Telecommunications Standards Institute (ETSI).

European Commission, 2008. Cars that talk: Commission earmarks single radio frequency for road safety and traffic management (No. IP/08/1240). Europa.eu.

European Parliament, Council of the European Union, 2010. Directive 2010/40/EU of the European Parliament and of the Council. Official Journal of the European Union 50, 207.

Evans, T.P., Kelley, H., 2004. Multi-scale analysis of a household level agent-based model of landcover change. Journal of Environmental Management 72, 57–72. doi:10.1016/j.jenvman.2004.02.008

Evans, T.P., Sun, W., Kelley, H., 2006. Spatially explicit experiments for the exploration of land-use decision-making dynamics. International Journal of Geographical Information Science 20, 1013–1037.

Fernandes, P., Nunes, U., 2007. Vehicle Communications: A Short Survey, in: IADIS International Conference on Telecommunications, Networks and Systems. Lisbon, Portugal, pp. 134–138.

Fiore, M., 2006. Mobility models in inter-vehicle communications literature. Politecnico di Torino.

Flache, A., Hegselmann, R., 2001. Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics. Journal of Artificial Societies and Social Simulation 4.

Fossett, M., 2006. Ethnic Preferences, Social Distance Dynamics, and Residential Segregation: Theoretical Explorations Using Simulation Analysis. Journal of Mathematical Sociology 30, 185–273.

Fremont, G., 2004. European DSRC Applications Developments. Presented at the 11th ITS World Congress, Special Session 27: Development of DSRC Multiple Application, Nagoya, Japan.

Galton, A., 2005. Dynamic Collectives and Their Collective Dynamics, in: Cohn, A., Mark, D. (Eds.), Spatial Information Theory, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 300–315.

Galton, A., Worboys, M., 2011. An Ontology of Information for Emergency Management, in: Proceedings of the 8th International ISCRAM Conference. Lisbon, Portugal.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., 2003. Sweetening WORDNET with DOLCE. AI Magazine 24, 13–24. doi:10.1609/aimag.v24i3.1715

Gasevic, D., Djuric, D., Devedzic, V., 2006. Model driven architecture and ontology development. Springer-Verlag.

Gáspár, P., Szalay, Z., Aradi, S., 2014. Highly Automated Vehicle Systems. BME MOGI.

Gearon, P., Passant, A., Polleres, A., 2012. SPARQL 1.1 Update. W3C Working Draft.

Genesereth, M.R., Nilsson, N.J., 1987. Logical foundations of artificial intelligence. Morgan Kaufmann Los Altos, CA.

Gerla, M., Lee, E.K., Pau, G., Lee, U., 2014. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds, in: 2014 IEEE World Forum on Internet of Things (WF-IoT). Presented at the 2014 IEEE World Forum on Internet of Things (WF-IoT), pp. 241–246. doi:10.1109/WF-IoT.2014.6803166

Gimblett, H.R., Richards, M.., Itami, R.M., 2002. Simulating wildland recreation use and conflicting spatial interactions using rule-driven intelligent agents, in: Integrating Geographic Information Systems and Agent-Based Modeling Techniques for Simulating Social and Ecological Processes. Oxford University Press, pp. 211–243.

Giorgini, P., Henderson-Sellers, B., 2005. Agent-oriented methodologies: an introduction, in: Agent-Oriented Methodologies. Idea Group Publishing, pp. 1–19.

Gómez-Pérez, A., Fernández-López, M., Corcho, O., 2004. Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Springer.

Goodchild, M.F., 2000. GIS and Transportation: Status and Challenges. GeoInformatica 4, 127–139. doi:10.1023/A:1009867905167

Goodchild, M.F., Egenhofer, M.J., Kemp, K.K., Mark, D.M., Sheppard, E., 1999. Introduction to the Varenius project. International Journal of Geographical Information Science 13, 731–745. doi:10.1080/136588199240996

Goodchild, M.F., Yuan, M., Cova, T.J., 2007. Towards a general theory of geographic representation in GIS. International Journal of Geographical Information Science 21, 239–260. doi:10.1080/13658810600965271

Gottfredson, L.S., 1997. Mainstream science on intelligence: An editorial with 52 signatories, history, and bibliography. Intelligence 24, 13–23.

Grant, J., Beckett, D., 2004. RDF Test Cases. W3C Recommendation.

Grenon, P., Smith, B., 2004. SNAP and SPAN: Towards Dynamic Spatial Ontology. Spatial Cognition & Computation 4, 69–104. doi:10.1207/s15427633scc0401_5

Gruber, T., 2009. Ontology, in: Liu, L., Özsu, M.T. (Eds.), Encyclopedia of Database Systems. Springer Publishing Company, Incorporated.

Gruber, T., 1993. A translation approach to portable ontology specifications. Knowledge acquisition 5, 199–220.

Guarino, N., 1998. Formal Ontology and Information Systems, in: Guarino, N. (Ed.), Formal Ontology in Information Systems: Proceedings of the First International Conference (FOIS'98). IOS Press, Trento, Italy, pp. 3–15.

Guarino, N., Giaretta, P., 1995. Ontologies and Knowledge Bases towards a Terminological Clarification, in: Mars, N.J.I. (Ed.), Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing. IOS Press, Amsterdam, Netherlands.

Haklay, M., O'Sullivan, D., Thurstain-Goodwin, M., Schelhorn, T., 2001. "So go downtown": simulating pedestrian movement in town centres. Environment and Planning B: Planning and Design 28, 343 – 359. doi:10.1068/b2758t

Hallé, S., Chaib-draa, B., 2005. A collaborative driving system based on multiagent modelling and simulations. Transportation Research Part C: Emerging Technologies 13, 320–345. doi:10.1016/j.trc.2005.07.004

Hansmann, U., Merk, L., Nicklous, M.S., Stober, T., 2003. What Pervasive Computing Is All About, in: Pervasive Computing: The Mobile World. Springer, pp. 11–24.

Härri, J., Filali, F., Bonnet, C., 2007. Mobility Models for Vehicular Ad Hoc Netw orks: A Survey and Taxonomy (Research Report), RR-06-168. Department of Mobile Communications, Eurecom.

Härri, J., Filali, F., Bonnet, C., Fiore, M., 2006. VanetMobiSim: generating realistic mobility patterns for VANETs, in: Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks. ACM, pp. 96–97.

Harris, S., Seaborne, A., 2012. SPARQL 1.1 Query Language. W3C Working Draft.

Helbing, D., Balietti, S., 2012. Agent-based Modeling, in: Helbing, D. (Ed.), Social Self-Organization - Agent-Based Simulations and Experiments to Study Emergent Social Behavior. Springer-Verlag Berlin Heidelberg, pp. 25–70.

Henchey, M.J., Batta, R., Blatt, A., Flanigan, M., Majka, K., 2013. A simulation approach to study emergency response. J Simulation 8, 115–128. doi:10.1057/jos.2013.20

Holfelder, W., 2004. Vehicle-to-Vehicle and Vehicle-to-Infrastructure Communication Recent Developments, Opportunities and Challenges, in: Automotive Software Workshop, Future Generation Software Architectures in the Automotive Domain: Connected Services in Mobile Networks. San Deigo, CA, USA.

Hornsby, K.S., King, K., 2008. Modeling Motion Relations for Moving Objects on Road Networks. GeoInformatica 12, 477–495. doi:10.1007/s10707-007-0039-7

Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C., 2011. A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3.

Horrocks, I., 2010. Description Logic: a Formal Foundation for Languages and Tools.

Horrocks, I., 2009. OWL 2: The Next Generation. London Semantic Web Meetup Group Seminar. London, UK.

Horrocks, I., 2008. Ontologies and the semantic web. Commun. ACM 51, 58–67. doi:10.1145/1409360.1409377

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M., 2004. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission.

Howard, D., Dai, D., 2014. Public Perceptions of Self-driving Cars: The Case of Berkeley, California. Presented at the The Transportation Research Board (TRB) 93rd Annual Meeting, Washington, D.C.

Hussain, R., Rezaeifar, Z., Oh, H., 2015. A Paradigm Shift from Vehicular Ad Hoc Networks to VANET-Based Clouds. Wireless Pers Commun 83, 1131–1158. doi:10.1007/s11277-015-2442-y

Hussain, R., Son, J., Eun, H., Kim, S., Oh, H., 2012. Rethinking Vehicular Communications: Merging VANET with cloud computing, in: 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom). Presented at the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 606–609. doi:10.1109/CloudCom.2012.6427481

Hutt, K., 2005. A comparison of RDF query languages, in: Proc. of 21th Computer Science Seminar. Hartfort, Connecticut.

Ipfelkofer, F., Lorenz, B., Ohlbach, H.J., 2006. Ontology Driven Visualisation of Maps with SVG - An Example for Semantic Programming, in: Tenth International Conference on Information Visualization (IV 2006). pp. 424–429. doi:10.1109/IV.2006.79

Israel, G.D., 2012. Determining Sample Size (No. PE0D6), UF/IFAS Extension. Electronic Data Information Source of UF/IFAS Extension, nstitute of Food and Agricultural Sciences (IFAS), University of Florida, Gainesville, FL 32611.

Jasper, R., Uschold, M., 1999. A framework for understanding and classifying ontology applications, in: Proceedings 12th Int. Workshop on Knowledge Acquisition, Modelling, and Management KAW. pp. 16–21.

Jennings, N.R., 2001. An agent-based approach for building complex software systems. Commun. ACM 44, 35–41. doi:10.1145/367211.367250

Jiang, D., Taliwal, V., Meier, A., Holfelder, W., Herrtwich, R., 2006. Design of 5.9 ghz dsrc-based vehicular safety communication. IEEE Wireless Communications Magazine 13, 36–43. doi:10.1109/WC-M.2006.250356

Johnston, K.M., North, M.J., Brown, D.G., 2013. Introducing agent-based modelling in the GIS environment, in: Johnston, K.M. (Ed.), Agent Analyst, Agent-Based Modeling in ArcGIS. Esri Press, Redlands, California, pp. 1–30.

Kargl, F., Ma, Z., Schoch, E., 2006. Security engineering for VANETs, in: 4th Workshop on Embedded Security in Cars (Escar 2006). Citeseer.

Karnadi, F.K., Mo, Z.H., Lan, K., 2007. Rapid generation of realistic mobility models for VANET, in: 2007 IEEE Wireless Communications and Networking Conference. IEEE, pp. 2506–2511.

Kerner, B., 1999. Congested Traffic Flow: Observations and Theory. Transportation Research Record: Journal of the Transportation Research Board 1678, 160–167. doi:10.3141/1678-20

Kesting, A., Treiber, M., Helbing, D., 2008. Agents for traffic simulation. arXiv preprint arXiv:0805.0300.

Khairnar, V.D., Pradhan, S.N., 2010. Mobility Models for Vehicular Ad-hoc Network Simulation. International Journal of Computer Applications 11, 8–12. doi:10.5120/1573-2103

Kifer, M., Boley, H., 2010. RIF Overview. W3C Working Group Note.

Klyne, G., Carroll, J.J., 2004. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation.

Knublauch, H., 2011. Update inferences. TopBraid Suite Users.

Knublauch, H., 2009. Magic Properties with SPIN. Composing the Semantic Web.

Knublauch, H., Hendler, J.A., Idehen, K., 2011. SPIN - Overview and Motivation.

Kompfner, P., 2010. e-Car - Connecting up the dots. Presented at the FITCE.be (Belgian Federation of Telecom Engineers of the EU) symposium, Interdisciplinary Institute for Broadband Technology (IBBT), Gent, Belgium.

Kravari, K., Bassiliades, N., 2015. A Survey of Agent Platforms. JASSS 18, 11.

Krikorian, Y.H., 1935. The Concept of Organization. The Journal of Philosophy 32, 119–126. doi:10.2307/2016073

Kuhn, W., 2001. Ontologies in support of activities in geographical space. International Journal of Geographical Information Science 15, 613–631. doi:10.1080/13658810110061180

Lee, D., Meier, R., 2007. Primary-Context Model and Ontology: A Combined Approach for Pervasive Transportation Services, in: Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops ’07). pp. 419–424. doi:10.1109/PERCOMW.2007.95

Lieberman, J., Singh, R., Goad, C., 2007. W3C Geospatial Ontologies. W3C Incubator Group Report.

Linz, P., 2001. An Introduction to Formal Languages and Automata. Jones & Bartlett Learning.

Lorenz, B., Ohlbach, H.J., Yang, L., 2005. Ontology of transportation networks. REWERSE Deliverable A1-D4.

Luck, M., McBurney, P., Shehory, O., Willmott, S., 2005. Agent technology: computing as interaction (a roadmap for agent based computing).

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., 2004. Mason: A new multi-agent simulation toolkit, in: Proceedings of the 2004 SwarmFest Workshop.

Macal, C.M., North, M.J., 2010. Tutorial on agent-based modelling and simulation. Journal of Simulation 4, 151–162. doi:10.1057/jos.2010.3

Maccubbin, R.P., Staples, B.L., Kabir, F., Lowrance, C.F., Mercer, M.R., Philips, B.H., Gordon, S.R., 2008. Intelligent Transportation Systems Benefits, Costs, Deployment, and Lessons Learned: 2008 Update. Research and Innovative Technology Administration (RITA), U.S. Department of Transportation (US DOT), Washington, DC, USA.

Mahajan, A., Potnis, N., Gopalan, K., Wang, A., 2006. Evaluation of mobility models for vehicular ad-hoc network simulations, in: IEEE International Workshop on Next Generation Wireless Networks (WoNGeN).

Malleson, N., 2010. Agent-Based Modelling of Burglary (PhD thesis). School of Geography, The University of Leeds.

Mamadolimova, A., Ambiah, N., Lukose, D., 2011. Modeling Islamic Finance Knowledge for Contract Compliance in Islamic Banking, in: König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R.J., Jain, L.C. (Eds.), Knowledge-Based and Intelligent Information and Engineering Systems, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 346–355. doi:10.1007/978-3-642-23854-3_37

Mangharam, R., Weller, D., Rajkumar, R., Mudalige, P., Bai, F., 2006. Groovenet: A hybrid simulator for vehicle-to-vehicle networks, in: 2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services. IEEE, pp. 1–8.

Mangharam, R., Weller, D.S., Stancil, D.D., Rajkumar, R., Parikh, J.S., 2005. GrooveSim: a topography-accurate simulator for geographic routing in vehicular networks, in: Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks. ACM, pp. 59–68.

Manikandan, A., Dhas, C.S.G., 2012. ANALYSIS OF MOBILITY MODELS FRAMEWORK FOR VEHICULAR AD HOC NETWORKS. International Journal of Scientific & Engineering Research 3.

Manley, E., Cheng, T., 2011. Multi-agent simulation of drivers reactions to unexpected incidents on urban road networks, in: The 19th GIS Research UK (GISRUK) Conference. Plymouth, UK.

Marshall, G., 1998. ontology. A Dictionary of Sociology.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., 2003. WonderWeb Deliverable D18, Ontology Library (final).

Matuszek, C., Cabral, J., Witbrock, M., DeOliveira, J., 2006. An introduction to the syntax and content of Cyc, in: Proceedings of the 2006 Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering. Stanford University, California, USA, pp. 44–49.

McCreary, D., Kelly, A., 2013. Making Sense of NoSQL: A guide for managers and the rest of us, 1 edition. ed. Manning Publications, Shelter Island.

McGuinness, D.L., Harmelen, F. van, 2004. OWL Web Ontology Language Overview. W3C Recommendation.

Melton, J., 2006. SQL, XQuery, and SPARQL: what’s wrong with this picture, in: Proc. XTech: “Building Web 2.0.” Amsterdam, The Netherlands.

Miaoulis, G., Michener, R.D., 1976. An Introduction to Sampling. Kendall/Hunt Publishing Company.

Michaels, C., Kelley, D., Sumner, R., Chriss, S., 2010. DSRC Implementation Guide: A guide to users of SAE J2735 message sets over DSRC. SAE International.

Millner, J., Hale, M., Thompson, B., Roberts, C., 2006. Global Positioning System Handbook: GPS Data Collection for Integration with Geographic Information Systems Standards, Specifications and Best Practice Field Guide. Department of Sustainability and Environment & Department of Primary Industries, The State of Victoria, Australia.

Mladenić, D., Škrjanc, M., Kenda, K., Moraru, A., Bradeško, L., Fortuna, B., Škraba, P., 2012. Semantic data streams and stream ontologies software (No. Deliverable D4. 5). ENVISION (Environmental Services Infrastructure with Ontologies) Consortium.

Mlinarsky, F., Onishi, H., 2012. Wireless Technology Assessment for Automotive Applications. Presented at the ITS America 22nd Annual Meeting & Exposition.

Moray, N., 2004. Ou sont les neiges d’antan? (Where are the snows of yesteryear?), in: Human Performance, Situation Awareness and Automation Conference, 2nd. Daytona Beach, Florida, USA, p. 4.

Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., 2009. OWL 2 Web Ontology Language Profiles. W3C Recommendation.

Nassar, L., Jundi, A., Golestan, K., Sattar, F., Karray, F., Kamel, M., Boumaiza, S., 2012. Vehicular Ad-hoc Networks(VANETs): Capabilities, Challenges in Context-Aware Processing and Communication Gateway, in: Kamel, M., Karray, F., Hagras, H. (Eds.), Autonomous and Intelligent Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 42–49.

Neches, R., Fikes, R.E., Finin, T., Gruber, T., Patil, R., Senator, T., Swartout, W.R., 1991. Enabling Technology for Knowledge Sharing. AI Magazine 12, 36. doi:10.1609/aimag.v12i3.902

NHS Information Centre, 2008. Ambulance Services, England 2007-08. NHS Information Centre, Part of the Government Statistical Service.

Nittel, S., Duckham, M., Kulik, L., 2004. Information Dissemination in Mobile Ad-Hoc Geosensor Networks, in: Egenhofer, M.J., Freksa, C., Miller, H.J. (Eds.), Geographic Information Science, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 206–222.

Nittel, S., Stefanidis, A., Cruz, I., Egenhofer, M., Goldin, D., Howard, A., Labrinidis, A., Madden, S., Voisard, A., Worboys, M., 2004. Report from the first workshop on geo sensor networks. SIGMOD Rec. 33, 141–144. doi:10.1145/974121.974146

Noronha, V., Church, R.L., 2002. Linear Referencing and Alternate expressions of Location for Transportation (Task Order 3021). Vehicle Intelligence and Transportation Analysis Laboratory, National Center for Geographic Information and Analysis, Santa Barbara, California, USA.

Noy, N.F., McGuinness, D.L., 2001. Ontology development 101: A guide to creating your first ontology.

Nundloll, V., Grace, P., Blair, G.S., 2011. The Role of Ontologies in Enabling Dynamic Interoperability - Springer. Presented at the International Conference of Distributed Applications and Interoperable Systems (DAIS) 2011, Reykjavik, Iceland, p. pp 179-193. doi:10.1007/978-3-642-21387-8_14

Oberle, D., Ankolekar, A., Hitzler, P., Cimiano, P., Sintek, M., Kiesel, M., Mougouie, B., Baumann, S., Vembu, S., Romanelli, M., Buitelaar, P., Engel, R., Sonntag, D., Reithinger, N., Loos, B., Zorn, H.-P., Micelli, V., Porzel, R., Schmidt, C., Weiten, M., Burkhardt, F., Zhou, J., 2007. DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). Web Semantics: Science, Services and Agents on the World Wide Web 5, 156–174. doi:10.1016/j.websem.2007.06.002

Ohmori, S., Yamao, Y., Nakajima, N., 2000. The future generations of mobile communications based on broadband access technologies. Communications Magazine, IEEE 38, 134–142. doi:10.1109/35.888267

Olariu, S., Eltoweissy, M., Younis, M., 2011. Towards autonomous vehicular clouds. ICST Transactions on Mobile Communications and Applications 11, e2. doi:10.4108/icst.trans.mca.2011.e2

Olariu, S., Hristov, T., Yan, G., 2013. The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds, in: Basagni, S., Conti, rco, Giordano, S., Stojmenovic, I. (Eds.), Mobile Ad Hoc Networking. John Wiley & Sons, Inc., pp. 645–700.

Ontotext, 2014. THE TRUTH ABOUT TRIPLESTORES, The Top 8 Things You Need to Know When Considering a Triplestore.

Ordnance Survey, 2011. OS MasterMap® Integrated Transport Network^{TM} Layer Getting started guide. Ordnance Survey, SOUTHAMPTON, United Kingdom,.

Padgham, L., Winikoff, M., 2003. Prometheus: A Methodology for Developing Intelligent Agents, in: Giunchiglia, F., Odell, J., Weiß, G. (Eds.), Agent-Oriented Software Engineering III. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 174–185.

Pan, J., 2008. A survey of network simulation tools: Current status and future developments.

Papageorgiou, G., Maimaris, A., 2012. Modelling, Simulation Methods for Intelligent Transportation Systems, in: Abdel-Rahim, A. (Ed.), Intelligent Transportation Systems.

Parker, D.C., Meretsky, V., 2004. Measuring pattern outcomes in an agent-based model of edge-effect externalities using spatial metrics. Agriculture, Ecosystems & Environment 101, 233–250. doi:10.1016/j.agee.2003.09.007

Pease, A., Niles, I., Li, J., 2002. The suggested upper merged ontology: A large ontology for the semantic web and its applications, in: Working Notes of the Association for the

Advancement of Artificial Intelligence (AAAI) - 2002 Workshop on Ontologies and the Semantic Web. Edmonton, Canada.

Pell, J.P., Sirel, J.M., Marsden, A.K., Ford, I., Cobbe, S.M., 2001. Effect of reducing ambulance response times on deaths from out of hospital cardiac arrest: cohort study. BMJ 322, 1385–1388. doi:10.1136/bmj.322.7299.1385

Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D., 2014. Context aware computing for the internet of things: A survey. IEEE Communications Surveys & Tutorials 16, 414–454.

Perry, M., Herring, J., 2012. OGC GeoSPARQL - A Geographic Query Language for RDF Data (OGC Implementation Standard No. OGC 11-052r4). Open Geospatial Consortium.

Pinyon Labs/Noblis, Inc., 2010. IntelliDrive Deployment Scenarios Workshop Scenarios and Supporting Material, in: IntelliDrive Operational Scenarios Operational Scenarios Workshop. Washington, DC, USA.

Polikoff, I., 2011. Comparing SPIN with RIF. Voyages of the Semantic Enterprise.

Porzel, R., Malaka, R., 2004. A task-based approach for ontology evaluation. Presented at the Workshop on Ontology Learning and Population at the 16th European Converence on Artificial Intelligence (ECAI 2004), Valencia, Spain.

Railsback, S.F., Lytinen, S.L., Jackson, S.K., 2006. Agent-based Simulation Platforms: Review and Development Recommendations. SIMULATION 82, 609–623. doi:10.1177/0037549706073695

Rand, W., Zellner, M., Page, S.E., Riolo, R., Brown, D.G., Fernandez, L., 2002. The complex interaction of agents and environments: An example in urban sprawl, in: Proceedings of Agent. pp. 149–161.

Raubal, M., Winter, S., Teβmann, S., Gaisbauer, C., 2007. Time geography for ad-hoc shared-ride trip planning in mobile geosensor networks. ISPRS Journal of Photogrammetry and Remote Sensing 62, 366–381. doi:10.1016/j.isprsjprs.2007.03.005

Reuter, A., Zipf, A., 2008. Geographic Information Science: Where Next?, in: Wilson, J.P., Fotheringham, A.S. (Eds.), The Handbook of Geographic Information Science. Blackwell Publishing Ltd, pp. 609–619.

Saha, A.K., Johnson, D.B., 2004. Modeling mobility for vehicular ad-hoc networks, in: Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks. ACM, pp. 91–92.

Schagrin, M., 2008. National VII Architecture – Data Perspective, in: Transportation Research Board 2008 Annual Meeting, Session 415. Washington, DC, USA.

Schelling, T.C., 1971. Dynamic models of segregation. The Journal of Mathematical Sociology 1, 143–186. doi:10.1080/0022250X.1971.9989794

Schnacke, D., 2004. Proposed Applications for 5.9Ghz DSRC in North America. Presented at the 11th ITS World Congress, Special Session 27: Development of DSRC Multiple Application, Nagoya, Japan.

Seaborne, A., 2004. RDQL - A Query Language for RDF. W3C Member Submission.

Seeberger, D., 2008. Frequency Allocation for ITS.

Semy, S.K., Pulvermacher, M.K., Obrst, L.J., 2004. Toward the Use of an Upper Ontology for U.S. Government and U.S. Military Domains: An Evaluation.

Sequeda, J., 2012. Introduction to: RDF vs XML - DATAVERSITY [WWW Document]. URL http://www.dataversity.net/introduction-to-rdf-vs-xml/ (accessed 11.1.16).

Seremeti, L., Goumopoulos, C., Kameas, A., 2009. Ontology-based modeling of dynamic ubiquitous computing applications as evolving activity spheres. Pervasive and Mobile Computing 5, 574–591. doi:10.1016/j.pmcj.2009.05.002

Sharpe, B., Hodgson, T., 2006. Technology Forward Look- Towards a Cyber Urban Ecology. Foresight Directorate, London, UK.

Shen, W., Lang, S.Y.T., Wang, L., 2005. iShopFloor: an Internet-enabled agent-based intelligent shop floor. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 35, 371–381. doi:10.1109/TSMCC.2004.843224

Silberg, G., Wallace, R., Matuszak, G., Plessers, J., Brower, C., Subramanian, D., 2012. Self-Driving Cars – The Next Revolution (White paper). KPMG.

Sill, S., Christie, B., Diephaus, A., Garretson, D., Sullivan, K., Sloan, S., 2011. Intelligent Transportation Systems (ITS) Standards Program Strategic Plan for 2011–2014 (No. FHWA-JPO-11-052). Research and Innovative Technology Administration (RITA), U.S. Department of Transportation (US DOT), Washington, DC, USA.

Smith, B., Mark, D.M., 1998. Ontology and geographic kinds, in: Proceedings of the 8th International Symposium on Spatial Data Handling (SDH'98). Vancouver, Canada.

Smith, B.W., Weiner, G., 2013. Automated Driving: Legislative and Regulatory Action [WWW Document]. The Center for Internet and Society at Stanford Law School. URL http://cyberlaw.stanford.edu/wiki/index.php/Automated_Driving:_Legislative_and_Regulatory_Action#cite_note-0 (accessed 8.21.13).

Sowa, J.F., 2000. Knowledge representation: logical, philosophical, and computational foundations. Brooks/Cole Pacific Grove, CA.

Staab, S., Studer, R., 2009. Preface, Overview, in: Staab, S., Studer, R. (Eds.), Handbook on Ontologies. Springer, pp. vii–xvi.

Stevenson, D., Anderson, I., Berwin, N., Heppell, S., Summers, N., Whatford, C., Winkley, D., 1997. Information and communications technology in UK schools An independent inquiry.

Strang, T., Linnhoff-Popien, C., 2004. A context modeling survey, in: Workshop on Advanced Context Modelling, Reasoning and Management. Nottingham, UK.

Stuckenschmidt, H., Ceri, S., Valle, E.D., Harmelen, F.V., 2010. Towards expressive stream reasoning, in: Semantic Challenges in Sensor Networks. Number 10042 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, Schloss Dagstuhl - Leibniz-Zentrum Fuer Informatik.

Studer, R., Benjamins, V.R., Fensel, D., 1998. Knowledge engineering: Principles and methods. Data & Knowledge Engineering 25, 161–197. doi:10.1016/S0169-023X(97)00056-6

Swartout, B., Patil, R., Knight, K., Russ, T., 1997. Toward distributed use of large-scale ontologies. Presented at the Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium, The AAAI Press, Menlo Park, California, USA, pp. 138–148.

The Assessment Operation Group, 2012. Guidelines for Using Confidence Intervals for Public Health Assessment. The Washington State Department of Health, U.S.A.

The European Commission's Information Society Technologies Advisory Group, 2005. Ambient Intelligence: From Vision to Reality, in: Ambient Intelligence. IOS Press, pp. 45–68.

The Futures Group, 1994. SCENARIOS. The Futures Group.

Torp-Pedersen, C., Birk-Madsen, E., Pedersen, A., 1989. The time factor in resuscitation initiated by ambulance drivers. Eur Heart J 10, 555–557.

Treiber, M., Kesting, A., 2013. Traffic Flow Dynamics. Springer Berlin Heidelberg, Berlin, Heidelberg.

US Department of Transportation, 2009. Intelligent Transportation Systems Standards Fact Sheet, IEEE 1609 - Family of Standards for Wireless Access in Vehicular Environments (WAVE). US Department of Transportation.

US Federal Communications Commission, 2006. FCC Report and Order 06-110: Amendment of the Commission's Rules Regarding Dedicated Short-Range Communication Services in the 5.850-5.925 GHz Band (5.9 GHz Band), Amendment of parts 2 and 90 of the

Commission's Rules to allocate the 5.850-5.925 GHz Band to the Mobile Service (No. FCC Report and Order 06-110). Federal Communications Commission (FCC).

US Federal Communications Commission, 2003. FCC Report and Order 03-324: Amendment of the Commission's Rules Regarding Dedicated Short-Range Communication Services in the 5.850-5.925 GHz Band (5.9 GHz Band), Amendment of parts 2 and 90 of the Commission's Rules to allocate the 5.850-5.925 GHz Band to the Mobile Service (No. FCC Report and Order 03-324). Federal Communications Commission (FCC).

US Federal Communications Commission, 1999. FCC allocates spectrum in 5.9, GHz range for intelligent transportation systems uses (No. Report No. ET 99-5). Federal Communications Commission News.

Valle, E.D., 2014. Introduction to Stream Reasoning. Presented at the Tutorial on Stream Reasoning for Linked Data, Collocated with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Trentino, Italy.

Valle, E.D., Ceri, S., Harmelen, F. van, Fensel, D., 2009. It's a Streaming World! Reasoning Upon Rapidly Changing Information. IEEE Intelligent Systems 24, 83–89. doi:10.1109/MIS.2009.125

Viriyasitavat, W., Bai, F., Tonguz, O.K., 2010. UV-CAST: an urban vehicular broadcast protocol, in: Vehicular Networking Conference (VNC), 2010 IEEE. IEEE, pp. 25–32.

W3C OWL Working Group, 2009. OWL 2 Web Ontology Language Document Overview. W3C Working Draft.

Wang, J., Ding, Z., Jiang, C., 2005. An Ontology-based Public Transport Query System, in: The First International Conference on Semantics, Knowledge and Grid (SKG '05). p. 62. doi:10.1109/SKG.2005.41

Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K., 2004. Ontology Based Context Modeling and Reasoning using OWL, in: Second IEEE Annual Conference on Pervasive Computing and Communications Workshops. Orlando, Florida, USA, pp. 18–22. doi:http://doi.ieeecomputersociety.org/10.1109/PERCOMW.2004.1276898

Winikoff, M., 2005. Jack^{TM} Intelligent Agents: An Industrial Strength Platform, in: Bordini, R.H., Dastani, M., Dix, J., Fallah Seghrouchni, A. (Eds.), Multi-Agent Programming. Springer-Verlag, New York, pp. 175–193.

Winter, S., Nittel, S., 2006. Ad hoc shared-ride trip planning by mobile geosensor networks. International Journal of Geographical Information Science 20, 899–916. doi:10.1080/13658810600816664

Wooldridge, M., Jennings, N.R., 1995. Agent theories, architectures, and languages: A survey, in: Intelligent Agents, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 1–39.

Wooldridge, M., Jennings, N.R., Kinny, D., 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3, 285–312. doi:10.1023/A:1010071910869

Wooldridge, M.J., 2009. An Introduction to MultiAgent Systems. John Wiley and Sons.

Worboys, M., 2005. Event-oriented approaches to geographic phenomena. International Journal of Geographical Information Science 19, 1–28. doi:10.1080/13658810412331280167

Worboys, M., Duckham, M., 2004. Models of geospatial information, in: GIS: A Computing Perspective. CRC Press, pp. 133–167.

Worboys, M., Hornsby, K., 2004. From Objects to Events: GEM, the Geospatial Event Model, in: Egenhofer, M.J., Freksa, C., Miller, H.J. (Eds.), Geographic Information Science, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 327–343.

Xu, Q., Mak, T., Ko, J., Sengupta, R., 2004. Vehicle-to-vehicle safety messaging in DSRC, in: First ACM International Workshop on Vehicular Ad Hoc Networks. New York, NY, USA, pp. 19–28. doi:10.1145/1023875.1023879

Yamane, T., 1967. Statistics: an introductory analysis. Harper & Row.

Yang, F., Wang, S., Li, J., Liu, Z., Sun, Q., 2014. An overview of Internet of Vehicles. China Communications 11, 1–15. doi:10.1109/CC.2014.6969789

Yang, L., Worboys, M., 2011. A Navigation Ontology for Outdoor-Indoor Space (Work-in-progress). Presented at the Third ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness (ISA 2011), Chicago, IL, USA.

Yin, J., ElBatt, T., Yeung, G., Ryu, B., Habermas, S., Krishnan, H., Talty, T., 2004. Performance evaluation of safety applications over DSRC vehicular ad hoc networks, in: First ACM International Workshop on Vehicular Ad Hoc Networks. New York, NY, USA, pp. 1–9. doi:10.1145/1023875.1023877

Yin-fei, D., Ying-yong, Z., Nian-feng, L., 2015. Research Overview on Vehicular Ad Hoc Networks Simulation." 8.3 (2015): . International Journal of Control and Automation 8, 207–216.

Zambonelli, F., Jennings, N., Wooldridge, M., 2001. Organisational Abstractions for the Analysis and Design of Multi-agent Systems, in: Ciancarini, P., Wooldridge, M. (Eds.), Agent-Oriented Software Engineering, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 407–422.

Zambonelli, F., Jennings, N.R., Wooldridge, M., 2003. Developing multiagent systems: The Gaia methodology. ACM Trans. Softw. Eng. Methodol. 12, 317–370. doi:10.1145/958961.958963

Zheng, H., Son, Y.-J., Chiu, Y.-C., Head, L., Feng, Y., Xi, H., Kim, S., Hickman, M., 2013. A Primer for Agent-Based Simulation and Modeling in Transportation Applications (No. FHWA-HRT-13-054). The Federal Highway Administration (FHWA).

Appendices

Appendix A - Dedicated Short-Range Communication (DSRC)

This research chooses DSRC as the physical layer of vehicular communication to describe specific traffic situations. On top of DSRC, this research focused on the data layer (i.e. context modelling and semantic contents for vehicular communications) and the application layer (i.e. implementation of intelligent and communicative vehicle agents) of vehicular communication. Even though this research was designed based on DSRC technology, the contents of this thesis were not tightly related to DSRC technology itself. Therefore, detailed explanations of DSRC were not included in the main contents.

This appendix describes DSRC capacity, possible contents, radio frequency allocation, related standards, and application-level support to complement the physical capability and capacity of DSRC. The capability and possible contents of DSRC are covered in Appendix A.1. Appendix A.2 outlines DSRC radio frequency and information exchange protocols, while Appendix A.3 describes possible applications of ITS via various communication modes, and the role of DSRC in safety-critical applications.

A.1 DSRC capability and its possible contents

Various wireless communication technologies can be applied to interconnect moving vehicles in order to create a Vehicular *Ad-hoc* Network (VANET) (Architecture Development Team, 2007; Kompfner, 2010). In this wireless *ad hoc* network, every participating vehicle is treated as a wireless router or node transmitting messages. To avoid traffic collisions for example, it is paramount that warning messages are sent using low-latency communication standards. Dedicated Short-Range Communication (DSRC) is one such standard, designed as a short to medium range (up to 1 km) wireless protocol in the 5.9 GHz band for fast moving vehicles (over 60 miles per hour) in order to support vehicle-to-infrastructure communication and vehicle-to-vehicle communication (Fernandes and Nunes, 2007).

There are various wireless communication technologies such as 3rd Generation of mobile communication (3G), Worldwide Interoperability for Microwave Access (WiMAX), Wireless Fidelity (WiFi), Infrared and, Dedicated Short Range Communication (DSRC) (Kompfner, 2010). Different types of communication can be appropriate to implement interactions among vehicles and infrastructure depending on situations. DSRC has better capabilities (low latency, high data transfer rate, and high mobility communications) compared to other wireless technologies in order to support communications of high-speed vehicles on the road (Table A.1).

Table A.1 - Comparison of DSRC to various wireless technologies (Dulmage et al., 2006)

	DSRC	Cellular	Wi-Fi	WiMAX
Data rate	3-27Mbps	< 2 Mbps	6-54Mbps	30 to 40 Mbps
Latency	< 50ms	Seconds	Seconds	Second
Range	< 1km	< 10km	< 100m	< 15km
Mobility	> 60 mph	> 60 mph	< 5mph	> 60 mph
Nominal Bandwidth	10MHz	< 3MHz	20MHz	3.5 Mhz - 10MHz
Operating Band	5.86 - 5.92GHz (ITS-RS)	800 - 900MHz, 1.8 - 1.9GHz	2.4GHz, 5.2GHz (ISM)	2.5GHz
IEEE standard	802.11p (DSRC), 1609 (WAVE)	N/A	802.11a	802.16e

DSRC is the most appropriate physical layer of communications for vehicles and infrastructure in an ITS environment, especially for safety-critical applications (Appendix A.2). The DSRC technology allows vehicles - via On-Board Equipment (OBE) - and roadside facilities - via Road-Side Equipment (RSE) - to send and receive messages to resolve potential problematic situations. For example, DSRC is used for warning a possible traffic collision or giving way to an emergency vehicle. If there is a traffic situation, the vehicles in the vicinity and the road infrastructure communicate with each other to understand the current situation in advance and subsequently perform an action that will resolve this situation.

Table A.2 summarises some possible DSRC communication content types (for vehicle-to-infrastructure, infrastructure-to-vehicle, and vehicle-to-vehicle), which guide the further development of relevant DSRC's semantic contents. For example, an ambulance driving to an emergency site could ask a traffic controller to turn traffic lights green to avoid hold-ups at red lights or intersections (vehicle-to-infrastructure communication). A signal controller with an RSE can send a stop sign violation warning to a specific vehicle which is in danger of running a red light (infrastructure-to-vehicle communication) (Holfelder, 2004). Additionally, vehicle-to-vehicle communications can be used to resolve other traffic situations. If there is an accident on the motorway and some vehicles that have an OBE are either involved in the accident or coming to the accident spot, the vehicles can send information back to other vehicles toward the scene by communication and turning on emergency lights. Recipient vehicles can relay the information back to other vehicles again. Therefore, vehicles towards the accident spot can reduce their speeds more safely and get ready for the accident spot in advance (Holfelder, 2004).

Table A.2 - Possible contents for DSRC (Schagrin, 2008)

Communication type	Collection, Distribution and Exchange of Data
Vehicle to Infrastructure	• Probe Data • Trip Path Data (opt-in) • Transaction Data (e.g. E-Payment)
Infrastructure to Vehicle	• Advisory Message Data (e.g. travel times, incident information, local signage) • Localised Map Data (e.g. detailed roadway geometry for signalised intersections, non-signalised intersections, road curve segments) • Signal Phase & Timing Data • Position Corrections • Transaction Data (e.g. E-Payment)
Vehicle to Vehicle	• Heartbeat Data (e.g. vehicle's position, speed, direction of travel, and size)

A.2 DSRC radio frequency and information exchange protocols

In the USA, the DSRC band (5.850 - 5.925 GHz) was allocated with seven 10 MHz channels which consist of one control channel (Ch 178) and six service channels (US Federal Communications Commission, 2006, 2003, 1999). As Figure A.1 shows, for the US DSRC allocation, Ch 172 is assigned for vehicle-to-vehicle communications, Ch 174 and Ch 176 are for middle-range services, Ch 180 and Ch 182 are for short-range services, and Ch 184 is for intersections services (US Federal Communications Commission, 2003; US Federal Communications Commission, 2006). Additionally, Ch 174/176 and Ch 180/182 may be combined to create two 20 MHz channels (Ch 175 and Ch 181). As the US DSRC allocation is compatible with Canadian DSRC allocation and Mexican DSRC allocation, it provides a single North-America-wide frequency band.

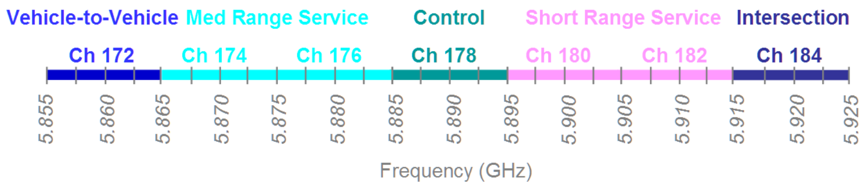


Figure A.1 - North American DSRC frequency allocation

Since 2008, European DSRC frequency has changed rapidly. The European Commission (EC) decided to allocate 30 MHz of the 5.875 – 5.905 GHz spectrum for DSRC to provide a single EU-wide frequency band (European Commission, 2008). Furthermore, the Electronic Communications Committee (ECC, 2008) of the European Conference of Postal and Telecommunications Administrations (CEPT) decided to use the 5.875 – 5.925 GHz band and recommended to use the 5.855 – 5.875 GHz band for 'the harmonised implementation of ITS'. For the harmonised European standard for ITS, the European Telecommunications Standards Institute (ETSI, 2008) is working under the Radio and Telecommunications Terminal Equipment (R&TTE) Directive Article 3.2, which is about effective use of the radio spectrum/orbital resource so as to avoid harmful interference.

- *ETSI EN 302 571 Candidate Harmonised Standard - Radiocommunications equipment operating in the 5.855 GHz to 5.925 GHz frequency band*

Consequently, the requested European frequency spectrum for DSRC was proposed in accordance with the North American DSRC allocation (Figure A.2). It can be divided into 20 Mhz, 30Mhz, and 20Mhz, as shown below (ETSI, 2006).

- *20 Mhz between 5.885 GHz and 5.905 GHz was allocated for Inter Vehicle (IVC) and Roadside to Vehicle (R2V) Communications providing critical road safety applications including the control channel*
- *30 MHz (between 5.875 GHZ and 5.885, and between 5.905 GHZ and 5.925GHZ) was for IVC and R2V providing road safety and traffic efficiency applications*
- *20 Mhz below 5.875 GHz was allocated for IVC and R2V providing non-safety-related applications*

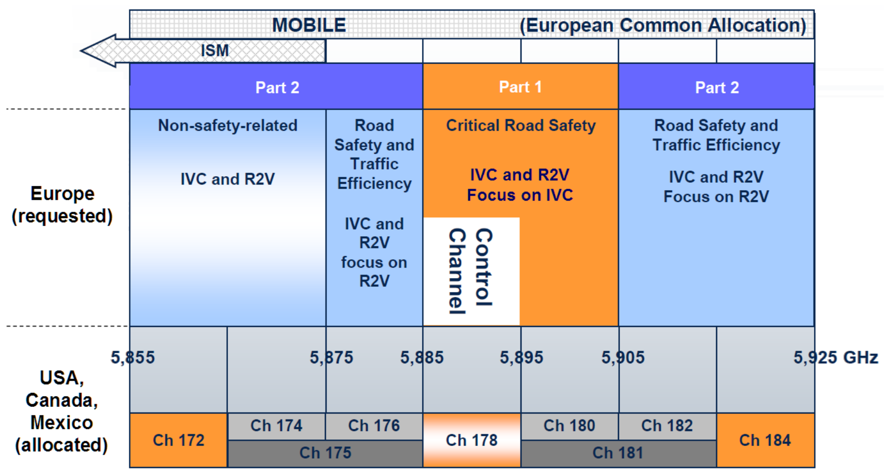


Figure A.2 - DSRC frequency in Europe and North America (edited from Seeberger, 2008)

DSRC has been developing based on the Institute of Electrical and Electronics Engineers (IEEE) 802.11p and IEEE 1609 standards. The former is a standard for Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications which include Wireless Access in Vehicular Environments (WAVE), and the latter was made as the family of standards for WAVE relying on the IEEE 802.11p including (US Department of Transportation, 2009):

- *ASTM 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*
- *IEEE 802.11p: Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part II: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*
- *IEEE 1609.4-2006: Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-Channel Operation*
- *IEEE 1609.3: Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services*
- *IEEE 1609.2-2006: Standard for Wireless Access in Vehicular Environments (WAVE) - Security Services for Applications and Management Messages*
- *IEEE 1609.1-2006: Standard for Wireless Access in Vehicular Environments (WAVE) - Resource Manager*

- *IEEE P1609.0: Standard for Wireless Access in Vehicular Environments (WAVE) - Architecture*

In Europe, the 5.795 – 5.815 GHz band has been used for DSRC, which is in the 5.725 GHz to 5.875 GHz Industrial, Scientific and Medical (ISM) band. The European Committee for Standardization (CEN) developed the following European DSRC standards in co-operation with the International Organisation for Standardization (ISO) (Fremont, 2004):

- *EN 12253:2004 DSRC Physical layer using microwave at 5.8 GHz*
- *EN 12795:2003 DSRC Data link layer: Medium Access and Logical Link Control*
- *EN 12834:2003 DSRC Application layer*
- *EN 13372:2004 DSRC profiles for Road Transport and Traffic Telematics (RTTT) applications*
- *EN ISO 15628: DSRC application layer*
- *EN ISO 14906:2004 Electronic Fee Collection - Application interface*

As we have seen so far, the 5.855 GHz to 5.925 GHz frequency band is considered for ITS application in North America and Europe, and this DSRC band can be a conventional and global frequency allocation. Even though this frequency is not allocated entirely yet in Europe, the DSRC band's capabilities (data rate, latency, range, mobility, etc.) have shown DSRC's potential to be a reliable communication technology for safety-related ITS applications.

A.3 DSRC in Vehicle Infrastructure Integration (VII) and ITS programs

The research to examine the potential of DSRC has been started in 2001 under the Vehicle Infrastructure Integration (VII) program to improve road safety, and current ITS program in the United States builds on top of research results of VII. Since 2009 ITS program focused on initial deployment considering all relevant communication technologies (Table A.3). Initially, VII program started only for road safety applications, but current ITS program needs to cover various ITS applications. Therefore, in the perspective of deployment and implementation there are many open questions about not only safety applications using DSRC but also mobility or environmental applications that may use other communication technologies like listed below (Sill et al., 2011).

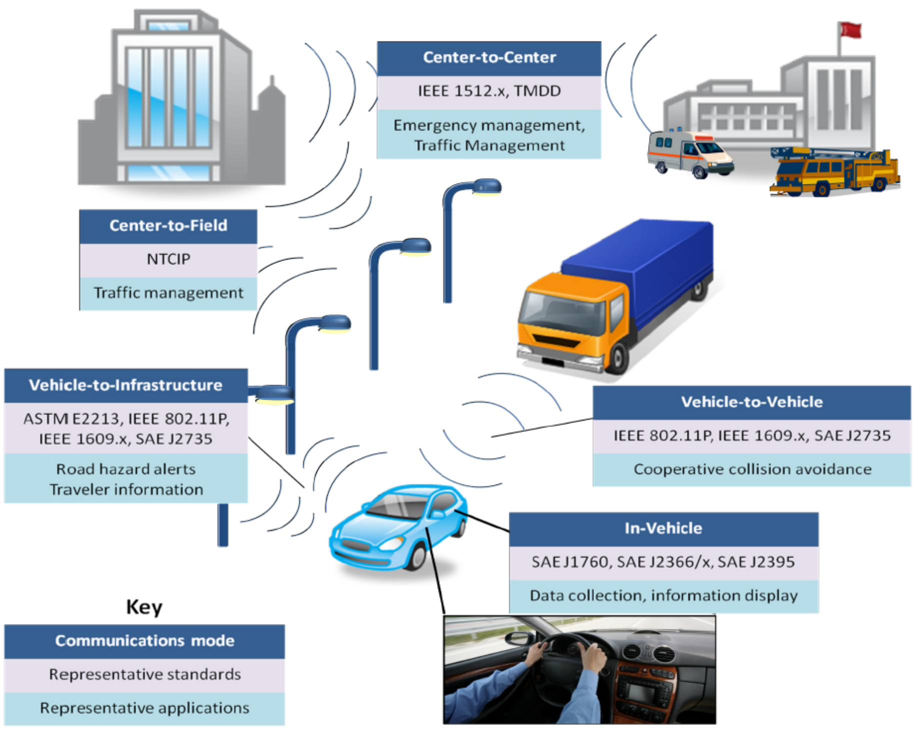
- *Are the safety benefits of near-term ITS applications sufficient to support National Highway Transportation Safety Administration (NHTSA) rules requiring DSRC-based safety devices in light and/or heavy vehicles?*
- *What additional non-safety (mobility or environmental) applications might encourage faster adoption of DSRC-based or other communications systems?*
- *How much and what type of roadside infrastructure is necessary to support safety and mobility applications?*

Despite these critical questions, DSRC is the only viable technology for low-latency safety applications currently (Sill et al., 2011).

Table A.3 - Changes from VII to today's ITS program in the United States (Sill et al., 2011)

Attribute	VII Engineering Research	ITS Program Focused Toward Deployment
Communications technologies	DSRC only	Best technology for intended application (DSRC for safety)
In-vehicle devices	OEM production units only	Aftermarket and retrofit opportunities
Vehicle Focus	Light vehicles	All vehicle types
Stakeholder involvement	Limited	Broad engagement
International focus	Limited	Significant international harmonization effort
Program cohesion	Loosely coupled research programs	Strong, collective U.S. Department of Transportation (USDOT) support, coordination, and leadership
Deployment focus	Limited – oriented toward prototyping and proof of concept	Strong deployment focus

Figure A.3 demonstrates possible standards (information exchange protocols and message definitions) and applications for various communication modes. When there is an emergency situation such as a car accident, a transport management centre may get the information about the accident from roadside infrastructure through centre-to-field communication. Then the transport manage centre will communicate with an emergency control centre (e.g. an ambulance command and control centre) to send the closest usable ambulance to the accident scene to provide aid. For vehicle-to-vehicle and vehicle-to-infrastructure communication modes, standards such as ASTM E2213, IEEE 802.11P, IEEE 1609.x, and SAE J2735 are presented in Figure A.3. First three standards are information exchange protocols, and fourth one is a standard for DSRC message definitions. These two communication modes are dealing with rapidly moving vehicles and most applications of these modes are related to safety-critical issues.



Center-to-Center
IEEE 1512.x, TMDD
Emergency management, Traffic Management
Center-to-Field
NTCIP
Traffic management
Vehicle-to-Infrastructure
ASTM E2213, IEEE 802.11P, IEEE 1609.x, SAE J2735
Road hazard alerts
Traveler information
Vehicle-to-Vehicle
IEEE 802.11P, IEEE 1609.x, SAE J2735
Cooperative collision avoidance
In-Vehicle
SAE J1760, SAE J2366/x, SAE J2395
Data collection, information display
Key
Communications mode
Representative standards
Representative applications

Figure A.3 - Promising ITS communication standards and applications (Sill et al., 2011)

Appendix B – Context modelling/representation techniques

Strang and Linnhoff-Popien (2004) and Perera et al. (2014) compared existing context modelling techniques (i.e. are key-value modelling, mark-up scheme modelling, graphical modelling, object oriented modelling, logic based modelling, and ontology based modelling) at a general level (Table A.4).

Table A.4 – Comparison of context modelling/representation techniques (Perera et al., 2014)

Technique	Pros	Cons
Key-Value	• Simple • Flexible • Easy to manage when small in size • Can be used for user preference, application configurations, limited data transferring	• Strongly coupled with applications • Not scalable • No structure or schema • Hard to retrieve information • No way to represent relationships • No validation support • No standard processing tools are available
Markup Scheme/ Tagged Encoding (e.g. XML, JSON)	• Flexible • More structured • Validation possible through schemas • Processing tools are available • Can be used for data description and transferring (e.g. XML, JSON)	• Application depended as there are no standards for structures • Can be complex when many levels of information are involved • Moderately difficult to retrieve information
Graphical (e.g. RDBMS, noSQL)	• Allows relationships modelling • Information retrieval is moderately easier • Different standards and implementations are available. • Validation possible through constraints • Can be used for long-term and large volume of permanent data archival in Relational DBMS • Structure alternation issue is no longer a problem in noSQL, and historic context can be stored in noSQL	• Querying can be complex • Configuration may be required • Interoperability among different implementation is difficult • No standards but governed by design principles
Object Based	• Allows relationships modelling • Can be well integrated using programming languages • Processing tools are available • Can be used to represent context in programming code level in computer memory	• Hard to retrieve information • No standards but govern by design principles • Lack of validation
Logic Based	• Allows to generate high-level context using low-level context • Simple to model and use • Support logical reasoning • Processing tools are available • Can be used to generate high-level context using low-level context, model events and actions (i.e. event detection), and define constrains and restrictions	• No standards • Lack of validation • Strongly coupled with applications

Ontology Based	• Support semantic reasoning • Allows more expressive representation of context and strong validation • Application independent and allows sharing • Strong support by standardisations • Fairly sophisticated tools available • Can be used to model domain knowledge and structure context based on the relationships defined by the ontology • Rather than storing data on ontologies, data can be stored in native triplestore, RDBMS-based triplestore, and NoSQL triplestore while structure is provided by ontologies	• Representation can be complex • Information retrieval can be complex and resource intensive

Appendix C - ANOVA to decide the number of simulation runs

Single factor ANOVA (a.k.a. one-factor ANOVA, one-way ANOVA) is used to compare four groups, which have statistics from different simulation runs. From the simulations, different variables (e.g. the number of vehicles affected by ambulance, the total affected time of vehicles by ambulance, and ambulance's travel time) are measured and used as a single independent variable for the analysis separately. Four different groups represent four different sizes of simulation runs (i.e. 100 simulation runs, 200 simulation runs, 400 simulation runs and 1100 simulation runs), and their average and variance are compared.

Summary and ANOVA results for the number of vehicles affected by ambulance, the total affected time of vehicles by ambulance, and ambulance's travel time are shown in Appendix B.1, Appendix B.2, and Appendix B.3, respectively. For each analysis, the null hypothesis (H0) is that means of the chosen variable from each group are similar, and 95% confidence level (significance level a = 0.05) is used. From analysis of three different variables, the hypothesis of each analysis is accepted based on the condition shown in Table A.5. Therefore, the minimum number of simulation runs (i.e. 100 simulation runs) is chosen among four different groups based on the ANOVA results that present the statistics of three different variables measured by four different simulation runs are similar.

Table A.5 - Condition to accept or reject for single factor ANOVA

If	Then
test statistic > critical value (i.e. F > F crit)	Reject the null hypothesis
test statistic < critical value (i.e. F < F crit)	Accept the null hypothesis
p value < a	Reject the null hypothesis
p value > a	Accept the null hypothesis

C.1 Single Factor ANOVA: the number of vehicles affected by ambulance

The null hypothesis for this ANOVA is that the average of the number of vehicles affected by ambulance is the same for all four groups. The statistical summaries are shown in Table A.6 and Figure A.4. The F-statistics in Table A.7 shows that the p value (0.526) is greater than the significance level a (0.05) and the F calculated value (0.744) is smaller than the F critical value (2.610), so that the null hypothesis is accepted.

Table A.6 - Summary of four groups with the number of vehicles affected by ambulance

Group	Count	Sum	Average	Variance
G100	100	17230	172.3	690.535
G200	200	34076	170.4	809.001
G400	400	67313	168.3	708.243
G1100	1100	185917	169.0	715.651

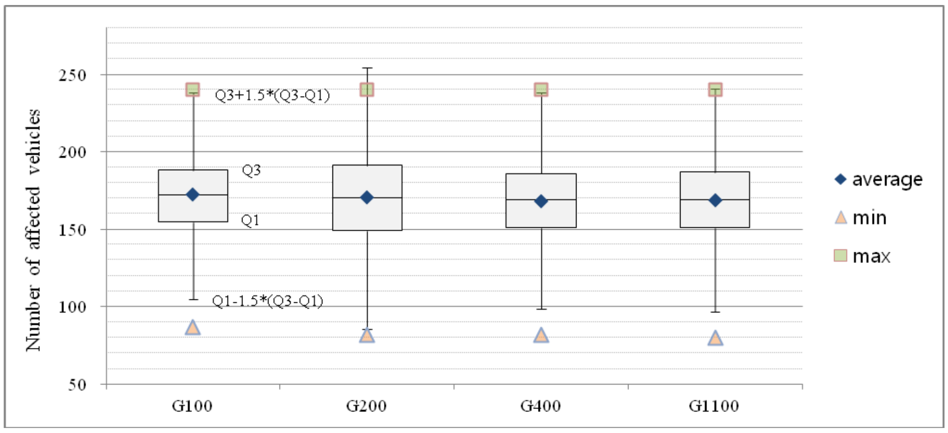


Figure A.4 - Boxplot of four groups with the number of affected vehicles

Table A.7 - F-statistics of the four groups with the number of vehicles affected by ambulance

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	1613.345	3	537.782	**0.744**	**0.526**	**2.610**
Within Groups	1298443.935	1796	722.964			
Total	1300057.280	1799				

C.2 Single Factor ANOVA: the total affected time of vehicles by ambulance

The null hypothesis for this ANOVA is that the average of the total affected time of vehicles by ambulance is the same for all four groups. The statistical summaries are shown in Table A.8 and Figure A.5. The F-statistics in Table A.9 shows that the p value (0.614) is greater than the significance level a (0.05) and the F calculated value (0.602) is smaller than the F critical value (2.610), so that the null hypothesis is accepted.

Table A.8 - Summary of four groups with the total affected time of vehicles by ambulance

Group	Count	Sum (sec)	Average (sec)	Variance
G100	100	57664.3	576.643	7854.844
G200	200	114027.6	570.138	9192.815
G400	400	225599.9	563.9998	8117.914
G1100	1100	623646.3	566.9512	8080.502

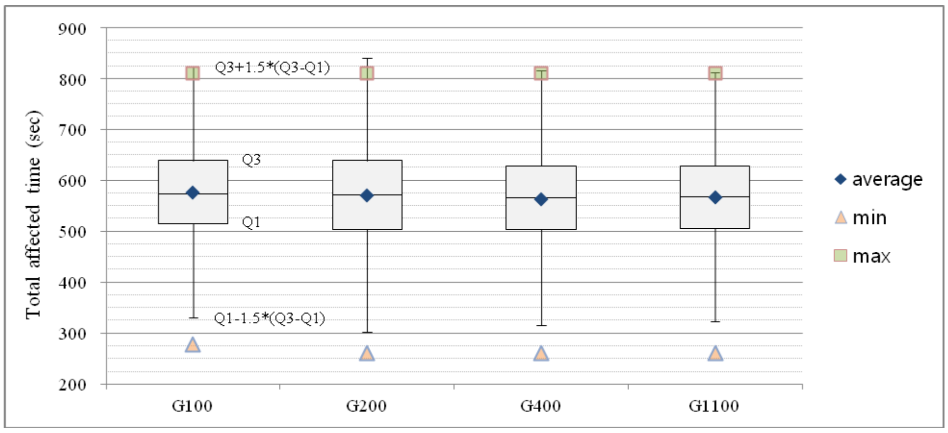


Figure A.5 - Boxplot of four groups with the total affected time of vehicles by ambulance

Table A.9 - F-statistics of the four groups with the total affected time of vehicles by ambulance

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	14807.869	3	4935.956	**0.602**	**0.614**	**2.610**
Within Groups	14726518.755	1796	8199.621			
Total	14741326.624	1799				

C.3 Single Factor ANOVA: ambulance's travel time

The null hypothesis for this ANOVA is that the average of ambulance's travel time is the same for all four groups. The statistical summaries are shown in Table A.10 and Figure A.6. The F-statistics in Table A.11 shows that the p value (0.567) is greater than the significance level a (0.05) and the F calculated value (0.676) is smaller than the F critical value (2.610), so that the null hypothesis is accepted.

Table A.10 - Summary of four groups with ambulance's travel time

Group	Count	Sum (sec)	Average (sec)	Variance
G100	100	29487.7	294.877	313.063
G200	200	58710.5	293.553	325.408
G400	400	116925.5	292.314	304.779
G1100	1100	322023.9	292.749	314.766

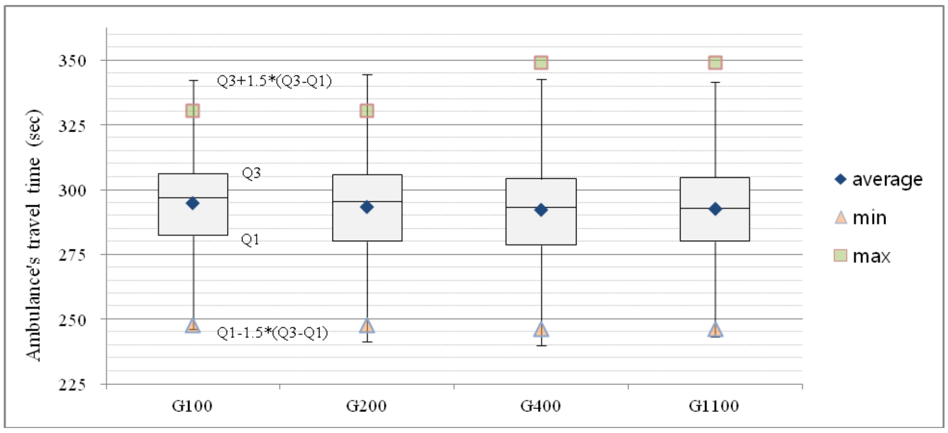


Figure A.6 - Boxplot of four groups with ambulance's travel time

Table A.11 - F-statistics of the four groups with ambulance's travel time

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	635.649	3	211.883	**0.676**	**0.567**	**2.610**
Within Groups	563284.379	1796	313.633			
Total	563920.028	1799				

Appendix D – Simulation source code

D.1 Parameters and simulation initialisation

D.1.1 GlobalVariables.java

```
public class GlobalVariables {

        public static String ROADS_SHAPEFILE;
        public static int iCase;

        public static final double TRAVEL_SPEED_MPH = 30.0; //dummy value
        public static final double TRAVEL_PER_TURN
                                                        = TRAVEL_SPEED_MPH * 1609.344 / 3600.0;
        public static final double ConversionCoefficient4MeterPerSec = 1609.344 / 3600.0;
        public static final boolean bVehicleInteraction = false;
        public static double commrange = 60.96; //200feet

        public static double defaultBreakdownResponseRange = 192.00; //4 seconds
        public static double extendedBreakdownResponseRange = 768.00; //16 seconds
        public static double searchRange4IDM = 576.00; //12 seconds

        public static int numVehicle;
        public static int numEmergencyVehicle;
        public static double MININUM_ROUTE_DISTANCE  = 3000;
        public static double COVERAGE_DISTANCE_FROM_HOSPITAL = 6000.0;
        public static int iPercentUsingComm;
        public static final double XXXX_BUFFER = 0.1; // Used in InfraContext.getRoadAtCoords()
        public static int  iRondomSeed;
        public static double endAt;

        public static int memorySizeForGabageColl = 6000;
        public static int printNum = 9;

        public static void setCase(int nCase) {
                iCase = nCase;

                if (nCase == 1) {
                        ROADS_SHAPEFILE
                   = "repast_vein_data/shape/roadLinks_PrincessRoyalUniversityHospitalSimple5.shp";
                } else if (nCase == 2) {
                        ROADS_SHAPEFILE
                         = "repast_vein_data/shape/roadLinks_UniversityCollegeHospitalSimple5.shp";
                } else if (nCase == 3) {
                        ROADS_SHAPEFILE
                                = "repast_vein_data/shape/roadLinks_KingstonHospitalSimple5.shp";
                } else if (nCase == 4) {
                        ROADS_SHAPEFILE
                                        = "repast_vein_data/shape/Polyline_MandAroad26.shp";
                        MININUM_ROUTE_DISTANCE  = 5000.0;
                }
        }
```

```
public static void StopScrolling (String str) {
        StopScrolling (true, str);
}

public static void StopScrolling (Boolean bCheck, String str) {
        System.out.println(str);
        if (!bCheck) return;
        System.out.print("Please, give me an int and press enter to continue:");
        BufferedReader consoleIn = new BufferedReader(new InputStreamReader(System.in));

        try {
                consoleIn.readLine();
        } catch (IOException e) {
                e.printStackTrace();
        }
}

public static void myPrint (String str) {
        myPrint(9, true, str);
}

public static void myPrint (int num, String str) {
        myPrint(num, true, str);
}

public static void myPrint (boolean bln, String str) {
        myPrint(9, bln, str);
}

public static void myPrint (int num, boolean bln, String str) {
        if (num <= printNum) {
                if (bln)
                        System.out.println (str);
                else
                        System.out.print(str);
        }
}

public static void GarbageCollector (String str, Boolean bStr) {
        long time = -1;
        long memTotal = -1;
        long memFree = -1;
        memTotal = Runtime.getRuntime().totalMemory() / 1024 / 1024 ;
        memFree = Runtime.getRuntime().freeMemory() / 1024 / 1024;
        if (memTotal > memorySizeForGabageColl) {
                time = System.currentTimeMillis();
                System.gc();
                if (bStr) {
                        System.out.println(str + "Garbage Collection - total memory: "
                                    + memTotal + "Mb, free memory: " + memFree + "Mb");
                        System.out.println(str + "It took "
                                          + (System.currentTimeMillis()-time) + " ms");
                }
        }
        if (memTotal > 7000) {
                 RunEnvironment.getInstance().endRun();
        }
```

```
		}
}
```

D.1.2 VeinContextCreator.java

```
public class VeinContextCreator implements ContextBuilder<Object> {

	private static int agentID; // Used to generate unique agent ids
	public static final boolean debug = false; // Turn debugging info on or off.
	private static double startTime;

	public Context<Object> build(Context<Object> context) {
		try {

		GlobalVariables.setCase(4);
		Parameters p = RunEnvironment.getInstance().getParameters();

		// Schedule simulation to stop at a certain time and also record the runtime.
		if (!VeinContextCreator.debug) {
			GlobalVariables.myPrint("Set Global Variables");
			GlobalVariables.numVehicle
						= (Integer) p.getValue("initialnumberofvehicles");
			GlobalVariables.numEmergencyVehicle
					= (Integer) p.getValue("initialnumberofambulances");
			GlobalVariables.commrange = (Double) p.getValue("communicationrange");
			GlobalVariables.iPercentUsingComm
							= (Integer) p.getValue("percentUsingOnt");
			GlobalVariables.endAt = (Double) p.getValue("runlength");
			RunEnvironment.getInstance().endAt(GlobalVariables.endAt);
			GlobalVariables.iRondomSeed = (Integer) p.getValue("randomSeed");
			RandomHelper.setSeed(GlobalVariables.iRondomSeed);
		}

		ContextOne context1 = new ContextOne();
		context.addSubContext(context1);

		GlobalVariables.myPrint("new Infra before");
		Infra infra = new Infra(context, "ContextOne", "RoadGeography", "RoadNetwork");
		infra.BuldingRoadNetwork();
		GlobalVariables.myPrint("new Infra after");
		GlobalVariables.myPrint("CreateEmergencyVehicles before");

		if (GlobalVariables.iCase == 4) {
			infra.CreateBrokenCar(GlobalVariables.numEmergencyVehicle);
		} else infra.CreateAmbulance(GlobalVariables.numEmergencyVehicle);

		PrivateVehicle.resetStaticVariables();
		GlobalVariables.myPrint("resetStaticVariables after");

		GlobalVariables.myPrint("CreatePrivateVehicles before");
		infra.CreatePrivateVehicles(GlobalVariables.numVehicle);
		GlobalVariables.myPrint("CreatePrivateVehicles after");
```

```
			ISchedule schedule = RunEnvironment.getInstance().getCurrentSchedule();
			ScheduleParameters endParams
				= ScheduleParameters.createAtEnd(ScheduleParameters.LAST_PRIORITY);
			schedule.schedule(endParams, this, "end");
			ScheduleParameters startParams = ScheduleParameters.createOneTime(1);
			schedule.schedule(startParams, this, "start");
			ScheduleParameters agentParams = ScheduleParameters.createRepeating(1, 1, 0);

			Iterable<Object> vehicleIt = context.getObjects(Vehicle.class);
			for (Object v:vehicleIt) {
				schedule.schedule(agentParams, v, "step");
					Vehicle ve = (Vehicle) v;
					ve.initialstep();
			}
			return context;

			} catch (Exception e) {
				GlobalVariables.StopScrolling("VeinContextCreator build before:" +  "\n"
										+ this.toString());
				e.printStackTrace();
				GlobalVariables.StopScrolling("VeinContextCreator build after:" +  "\n"
										+ this.toString());
			}
			return context;
		}

		public static void end() {
			GlobalVariables.myPrint(0, "Finished sim: "+(System.currentTimeMillis()-startTime));
		}
		public static void start() {
			System.setProperty("org.geotools.referencing.forceXY", "true");
			startTime = System.currentTimeMillis();
		}

		public static int generateAgentID() {
			return VeinContextCreator.agentID++;
		}
}
```

D.1.3 Infra.java

```
public class Infra {

		protected Context<Object> maincontext;
		private Geography roadGeography;
		private Network roadNetwork;
		protected String subcontextString, geographyString, networkString;
		private Network commNetwork;
		private DecimalFormat myFormatter;
		private int iVNum;
		private Junction fixedDestJunc;

		public String formatStr(double value) {
        return myFormatter.format(value);
}
public static String formatStr2(double value) {
        return new DecimalFormat("###000.000").format(value);
}
public Infra(Context<Object> mainContext, String subcontextString, String geograpyString,
                                                                String networkString) {
        myFormatter = new DecimalFormat("###000.000");
        this.maincontext = mainContext;
        this.subcontextString = subcontextString;
        this.geographyString = geograpyString;
        this.networkString = networkString;
        this.iVNum = 0;
        this.fixedDestJunc = null;
        if (VeinContextCreator.debug)
                GlobalVariables.myPrint("InfraContext: Building infra context");

        GeographyParameters geoParams = new GeographyParameters();
        geoParams.setCrs("EPSG:27700"); //http://spatialreference.org/ref/epsg/27700/
        roadGeography = GeographyFactoryFinder.createGeographyFactory(null)
                .createGeography(geograpyString, this.getRoadContext(), geoParams);
}
public void removeBrokenCar() {
        String name = "BrokenCar";
        Iterable<Object> emergencyVehicleIt
                        = this.getRoadContext().getObjects(EmergencyVehicle.class);
        for (Object o : emergencyVehicleIt) {
                EmergencyVehicle emergencyVehicle = (EmergencyVehicle) o;
                if (emergencyVehicle.getVehicleType().equals(name)) {
                        this.getRoadContext().remove(emergencyVehicle);
                }
                emergencyVehicle = null;
        }
}
public void CreateBrokenCar (int nBrokenCar) {
        Route refRoute = null;
        for (int i = 0; i < nBrokenCar; i++) {
                EmergencyVehicle brokenCar
                                = new EmergencyVehicle(this, "BrokenCar", true);
                brokenCar.setId(i);
                brokenCar.setName("E_B" + i);
                double dspeed = 69.0; //dummy value
                brokenCar.setSpeed(dspeed);
                this.getRoadContext().add(brokenCar);
                brokenCar.setOriginJuncRandomlyFromRandomHelper(4);
                brokenCar.setDestJunc(null);
                brokenCar.initializeRouteFromOriginFromRandomHelper(brokenCar
                   .getOriginJunc(), 4, GlobalVariables.MININUM_ROUTE_DISTANCE);
                brokenCar.updateRelativeLocation();
                brokenCar.setBroken(true);
                brokenCar.setPreferedLane(1);
                brokenCar.setEndTick(10*60*20);
                refRoute = brokenCar.route;
                double directDistance = brokenCar.getOriginJunc().getCoordinate().distance(
                                        brokenCar.getDestJunc().getCoordinate());
                GlobalVariables.COVERAGE_DISTANCE_FROM_HOSPITAL
                                                        = directDistance * 1.2;
        }
        MinimiseMotorway(refRoute, 1.0);
        EmergencyVehicle brokenCar = this.getAnEmergencyVehicle("BrokenCar");
        Vehicle bCar = (Vehicle) brokenCar;
        Coordinate myCoord = bCar.initialstep();
        MinimiseMotorway (myCoord, 2000.0); //2000.0
        brokenCar.setOriginJuncRandomlyFromRandomHelper(4);
        brokenCar.initializeRouteFromOriginFromRandomHelper(
                                        brokenCar.getOriginJunc(), 4, 4000);
        brokenCar.updateRelativeLocation();
}

public void BuldingRoadNetwork() {
        System.out.println("LoadRoadElement before");
        LoadRoadElement();
        System.out.println("CreateRoadNetwork before");
        CreateRoadNetwork(this.networkString);
        System.out.println("BuildRoadNetwork before");
        BuildRoadNetwork();
        GlobalVariables.myPrint("BuildRoadNetwork after");
}

public void CreateAmbulance (int nAmbulance) {
        Route refRoute = null;
        for (int i = 0; i < nAmbulance; i++) {
                EmergencyVehicle ambulance
                                = new EmergencyVehicle(this, "Ambulance", true);
                ambulance.setId(i);
                ambulance.setName("E_A" + i);
                double dspeed = GlobalVariables.TRAVEL_SPEED_MPH * 5.0 / 3.0;
                double speed = dspeed; // random(ispeed);
                ambulance.setSpeed(speed);
                ambulance.setPreferedLane(1);
                this.getRoadContext().add(ambulance);

                ambulance.setDestJunc(this.getFixedDestJunc());
                ambulance.setBroken(false);
                ambulance.setEndTick((int) GlobalVariables.endAt);
                ambulance.initializeRoute(null, ambulance.getDestJunc(), null, 0);
                ambulance.updateRelativeLocation();
                refRoute = ambulance.route;

                double directDistance = ambulance.getOriginJunc().getCoordinate().distance(
                                        ambulance.getDestJunc().getCoordinate());
                GlobalVariables.COVERAGE_DISTANCE_FROM_HOSPITAL
                                                = directDistance * 1.2;
        }
        if (GlobalVariables.iCase >= 1 && GlobalVariables.iCase <= 3) {
                MinimiseRoadNetwork (refRoute);
        }
}

public void RemoveRoadNetwork () {
        int junctionSize = this.getRoadContext().getObjects(Junction.class).size();
        Iterable<Object> junctionIt1 = this.getRoadContext().getObjects(Junction.class);
        if (junctionIt1 != null) {
                for (Object o : junctionIt1) {
                    Junction j = (Junction) o;
                    this.getRoadContext().remove(o);
                    j = null;
                }
            }

            Iterable<Object> roadElementIt1
                                = this.getRoadContext().getObjects(RoadElement.class);
            if (roadElementIt1 != null) {
                for (Object o : roadElementIt1) {
                    RoadElement r = (RoadElement) o;
                    this.getRoadContext().remove(r);
                    r = null;
                }
            }
            Iterable<Object> edgeIt1 = this.roadNetwork.getEdges();
            for (Object o : edgeIt1) {
                if (o instanceof RepastEdge) {
                    RepastEdge edge = (RepastEdge) o;
                    this.roadNetwork.removeEdge(edge);
                }
            }
            this.roadNetwork = null;
            this.commNetwork = null;
        }

        public void MinimiseRoadNetwork (Route refRoute) {
            Iterable<Object> roadElementIt
                                = this.getRoadContext().getObjects(RoadElement.class);
            List roadElements = new ArrayList();
            for (Object o : roadElementIt) {

                if (o instanceof RoadElement) {
                    RoadElement r = (RoadElement) o;
                    roadElements.add(r);
                }
            }
            Coordinate[] routeCoordinates
                = (Coordinate []) refRoute.getRouteCoords().toArray(new Coordinate[0]);
            LineString routeLine = new GeometryFactory().createLineString(routeCoordinates);
            Geometry routeBuffer = null;
            routeBuffer = routeLine.buffer(GlobalVariables.commrange * 5);

            for (Object o : roadElements) {
                if (o instanceof RoadElement) {
                    RoadElement r = (RoadElement) o;
                    Geometry roadGeom = roadGeography.getGeometry(r);
                    if (!routeBuffer.covers(roadGeom)) {
                        Junction fromJ = r.getFromJunc();
                        Junction toJ = r.getToJunc();
                    MyRepastEdge myEdge
                      = (MyRepastEdge) this.getRoadNetwork().getEdge(fromJ, toJ);
                        if (myEdge != null) {
                            this.roadNetwork.removeEdge(myEdge);
                        }
                        myEdge = (MyRepastEdge) this.getRoadNetwork()
                                                            .getEdge(toJ, fromJ);
```

```
                                        if (myEdge != null) {
                                                this.roadNetwork.removeEdge(myEdge);
                                        }

                                        fromJ.getConnectedRoads().remove(r);
                                        if (fromJ.getConnectedRoads().size() == 0)
                                                this.getRoadContext().remove(fromJ);
                                        toJ.getConnectedRoads().remove(r);
                                        if (toJ.getConnectedRoads().size() == 0)
                                                this.getRoadContext().remove(toJ);
                                        this.getRoadContext().remove(r);
                                }
                        }
                }

                Iterable<Object> roadEdgeIt = this.roadNetwork.getEdges();
                double totalNetworkLength = 0.0;
                for (Object o : roadEdgeIt) {
                        MyRepastEdge myEdge = (MyRepastEdge) o;
                        RoadElement road = getRoadElementWithID (myEdge.getRoadID());
                        totalNetworkLength += road.getShape_Leng(); //myEdge.getWeight();
                }
                GlobalVariables.myPrint("roadnetwork buffering end " + totalNetworkLength);
                GlobalVariables.numVehicle
                                = getNumOfVehiclesFromTotalNetworkLength(totalNetworkLength);
        }

        public int getNumOfVehiclesFromTotalNetworkLength(double totalNetworkLength) {
        //23 meters can be seen as six car lengths
        //so here, uses seven car lengths by 23 / 6 * 7.
        //since the road network for the simulation is buffered by the ambulance's route+commrange*5,
        //the network has a very long shape, and most vehicles come to the ambulance's route.
        //So, reduce the density of the main road, multiply 0.25 as experimental calculation
                int numVehicle = 0;
                if (GlobalVariables.iCase < 4)
                        numVehicle = (int) (totalNetworkLength / (23.0 / 6.0 * 7.0 * 4.0)); //30mph
                else if (GlobalVariables.iCase == 4) numVehicle =
                    (int) (totalNetworkLength / (96.0 / 24.0 * 25.0 * 4.0) * 3.0); //70mph, three lanes
                return numVehicle;
        }

        public void MinimiseMotorway (Route refRoute, double dist) {
                Iterable<Object> roadElementIt
                                        = this.getRoadContext().getObjects(RoadElement.class);
                List roadElements = new ArrayList();
                for (Object o : roadElementIt) {
                        if (o instanceof RoadElement) {
                                RoadElement r = (RoadElement) o;
                                roadElements.add(r);
                        }
                }
                Coordinate[] routeCoordinates
                        = (Coordinate []) refRoute.getRouteCoords().toArray(new Coordinate[0]);
                LineString routeLine = new GeometryFactory().createLineString(routeCoordinates);
                Geometry routeBuffer = null;
                routeBuffer = routeLine.buffer(dist); //(1.0);
```

```
		boolean bOutOfRoute;
		for (Object o : roadElements) {
			if (o instanceof RoadElement) {
				RoadElement r = (RoadElement) o;
				Geometry roadGeom = roadGeography.getGeometry(r);
				bOutOfRoute = false;
				if (routeBuffer.distance(roadGeom) <= dist) {
					if (refRoute.getRouteJunctions().contains(r.getFromJunc()) ||
					      refRoute.getRouteJunctions().contains(r.getToJunc()) )
										bOutOfRoute = false;
					else bOutOfRoute = true;
				} else if (routeBuffer.distance(roadGeom) > dist) {
					bOutOfRoute = true;
				}
				if (bOutOfRoute) {
					Junction fromJ = r.getFromJunc();
					Junction toJ = r.getToJunc();
					MyRepastEdge myEdge = (MyRepastEdge)
						this.getRoadNetwork().getEdge(fromJ, toJ);
					if (myEdge != null) {
						this.roadNetwork.removeEdge(myEdge);
					}
					myEdge = (MyRepastEdge)
						this.getRoadNetwork().getEdge(toJ, fromJ);
					if (myEdge != null) {
						this.roadNetwork.removeEdge(myEdge);
					}
					fromJ.getConnectedRoads().remove(r);
					if (fromJ.getConnectedRoads().size() == 0)
							this.getRoadContext().remove(fromJ);
					toJ.getConnectedRoads().remove(r);
					if (toJ.getConnectedRoads().size() == 0)
						this.getRoadContext().remove(toJ);
					this.getRoadContext().remove(r);
				}
			}
		}
		Iterable<Object> roadEdgeIt = this.roadNetwork.getEdges();
		double totalNetworkLength = 0.0;
		for (Object o : roadEdgeIt) {
			MyRepastEdge myEdge = (MyRepastEdge) o;
			RoadElement road = getRoadElementWithID (myEdge.getRoadID());
			totalNetworkLength += road.getShape_Leng(); //myEdge.getWeight();
		}
		GlobalVariables.numVehicle
			= getNumOfVehiclesFromTotalNetworkLength(totalNetworkLength);
	}

	public void MinimiseMotorway (Coordinate centerCoord, double refDistance) {
		Iterable<Object> roadElementIt
			= this.getRoadContext().getObjects(RoadElement.class);
		List roadElements = new ArrayList();
		for (Object o : roadElementIt) {
			if (o instanceof RoadElement) {
				RoadElement r = (RoadElement) o;
				roadElements.add(r);
			}
		}
		Point originPoint = new GeometryFactory().createPoint(centerCoord);
		for (Object o : roadElements) {
			if (o instanceof RoadElement) {
				RoadElement r = (RoadElement) o;
				Geometry roadGeom = roadGeography.getGeometry(r);
				if (originPoint.distance(roadGeom) > refDistance) {
					Junction fromJ = r.getFromJunc();
					Junction toJ = r.getToJunc();
					MyRepastEdge myEdge = (MyRepastEdge)
						this.getRoadNetwork().getEdge(fromJ, toJ);
					if (myEdge != null) {
						this.roadNetwork.removeEdge(myEdge);
					}
					myEdge = (MyRepastEdge)
						this.getRoadNetwork().getEdge(toJ, fromJ);
					if (myEdge != null) {
						this.roadNetwork.removeEdge(myEdge);
					}

					fromJ.getConnectedRoads().remove(r);
					if (fromJ.getConnectedRoads().size() == 0)
						this.getRoadContext().remove(fromJ);
					toJ.getConnectedRoads().remove(r);
					if (toJ.getConnectedRoads().size() == 0)
						this.getRoadContext().remove(toJ);
					this.getRoadContext().remove(r);
				}
			}
		}
		Iterable<Object> roadEdgeIt = this.roadNetwork.getEdges();
		double totalNetworkLength = 0.0;
		for (Object o : roadEdgeIt) {
			MyRepastEdge myEdge = (MyRepastEdge) o;
			RoadElement road = getRoadElementWithID (myEdge.getRoadID());
			totalNetworkLength += road.getShape_Leng(); //myEdge.getWeight();
		}
		GlobalVariables.myPrint("roadnetwork buffering end " + totalNetworkLength);
		GlobalVariables.totalNetworkLength = totalNetworkLength;
		GlobalVariables.numVehicle
				= getNumOfVehiclesFromTotalNetworkLength(totalNetworkLength);
	}

	public void CreatePrivateVehicles(int nPrivate) {
		/* add agents to the context and geography */
		for (int i = 0; i < nPrivate; i++) {

			boolean  bUseOnt = false;
			int iUseOnt = random(0, 100, false); //true

			if (iUseOnt < GlobalVariables.iPercentUsingComm) bUseOnt = true;
			else if (iUseOnt == 100 && GlobalVariables.iPercentUsingComm == 100)
										bUseOnt = true;
			PrivateVehicle pVehicle
					= new PrivateVehicle(this, "PrivateVehicle", bUseOnt);
			pVehicle.setId(this.getNextVehicleNumber());
			pVehicle.setName("V_" + pVehicle.getId());
			int iSpeedRandom = random(0, 100, false);
			pVehicle.setSpeedAndPreferedLane(iSpeedRandom);
			this.getRoadContext().add(pVehicle);
			if (GlobalVariables.iCase == 4)
				pVehicle.initializeRouteFromRandomHelper(null, null, null, 4);
			else pVehicle.initializeRoute(null, null, null);
			pVehicle.updateRelativeLocation();
		}
	}

	public void CreateAmbulanceAndPrivateVehicles(int nPrivate, int nAmbulance) {
		Route refRoute = null;
		for (int i = 0; i < nAmbulance; i++) {
			EmergencyVehicle ambulance
					= new EmergencyVehicle(this, "Ambulance", true);
			ambulance.setId(i);
			ambulance.setName("E_A" + i);
			double dspeed = GlobalVariables.TRAVEL_SPEED_MPH * 1.5; // 3.5;
			double speed = dspeed; // random(ispeed);
			ambulance.setSpeed(speed);
			this.getRoadContext().add(ambulance);
			ambulance.setDestJunc(this.getFixedDestJunc());
			ambulance.initializeRoute(null, ambulance.getDestJunc(), null);
			ambulance.updateRelativeLocation();
			refRoute = ambulance.route;

			double directDistance = ambulance.getOriginJunc().getCoordinate()
						.distance(ambulance.getDestJunc().getCoordinate()
						);
			GlobalVariables.COVERAGE_DISTANCE_FROM_HOSPITAL
									= directDistance * 1.2;
		}

		for (int i = 0; i < nPrivate; i++) {
			boolean  bUseOnt = false;
			int iUseComm = random (0, 100, false); //true
			if (iUseComm < GlobalVariables.iPercentUsingComm) bUseOnt = true;
			else if (iUseComm == 100 && GlobalVariables.iPercentUsingComm == 100)
										bUseOnt = true
			PrivateVehicle pVehicle
					= new PrivateVehicle(this, "PrivateVehicle", bUseOnt);
			pVehicle.setId(i);
			pVehicle.setName("V_" + i);
			double dspeed = GlobalVariables.TRAVEL_SPEED_MPH ; // 3.5;
			int add = random (-30, 30, false);
			double speed = dspeed  + dspeed * add / 100.0;
			pVehicle.setSpeed(speed);
			this.getRoadContext().add(pVehicle);
			pVehicle.setOriginJuncRandomly();
			pVehicle.initializeRoute(pVehicle.getOriginJunc(), null, refRoute);
			pVehicle.updateRelativeLocation();
		}
	}

	public void CreateRoadNetwork(String networkString) {
		NetworkFactory netFac = NetworkFactoryFinder
				.createNetworkFactory(new HashMap<String, Object>());
		roadNetwork = netFac.createNetwork(networkString, this.getRoadContext(), true);
		NetworkBuilder builder
			= new NetworkBuilder("CommNetwork", this.getRoadContext(), true);
		commNetwork = builder.buildNetwork();
	}

	/*
	 * Runs through all the junctions in the context. If it finds one with
	 * coordinates which are the same as the Junction passed to this functions
	 * it returns true.
	 */
	public boolean existsInContext(Junction j) {
		Iterable<Object> junctionIt = this.getRoadContext().getObjects(Junction.class);
		for (Object o : junctionIt) {
			if (o instanceof Junction) {
				Junction oj = (Junction) o;
				if (oj.equals(j)) {// o instanceof Junction &&
					return true;
				}
			}
		}
		return false;
	}

	public Junction getJunctionWithCoordinates(Coordinate c) {
		Iterable<Object> junctionIt = this.getRoadContext().getObjects(Junction.class);
		for (Object o : junctionIt) {
			if (o instanceof Junction) {
				Junction j = (Junction) o;
				if (j.getCoordinate().equals(c)) {
					return j;
				}
			}
		}
		System.err.print("Context1: getJunctionWithCoordinates: error, junction not found. ");
		System.err.println("Coordinates: " + c.toString());
		return null;
	}

	public void LoadRoadElement() {
		LoadRoadElement(null);
	}
	public void LoadRoadElement(String filename) {

		if (VeinContextCreator.debug)
			GlobalVariables.myPrint("Infra: building road context and projections");

		File roadFile = null;
		ShapefileLoader<RoadElement> roadLoader = null;

		try {
			if (filename == null)
				roadFile = new File(GlobalVariables.ROADS_SHAPEFILE);
			else roadFile = new File(filename);
			roadLoader = new ShapefileLoader<RoadElement>(RoadElement.class,
					roadFile.toURL(), roadGeography, this.getRoadContext());
			while (roadLoader.hasNext()) {
				RoadElement road = (RoadElement) roadLoader.next();
			}
			boolean bFilter = false;
			if (bFilter) {
				Iterable<Object> roadElementIt
					= this.getRoadContext().getObjects(RoadElement.class);
			List roadElements = new ArrayList();
			for (Object o : roadElementIt) {
				if (o instanceof RoadElement) {
					RoadElement r = (RoadElement) o;
					roadElements.add(r);
				}
			}
			for (Object o : roadElements) {
				RoadElement road = (RoadElement) o;
				GlobalVariables.myPrint(road.toString2());
				//for s2 only motorway and A-Road
				if (road.getDescTerm().equals("Pedestrianised Street") ||
					road.getDescTerm().equals("Alley") ||
					road.getDescTerm().equals("Local Street") ||
					road.getDescTerm().equals("B Road") ||
					road.getDescTerm().equals("Minor Road") ||
					road.getDescTerm().equals(
							"Private Road - Publicly Accessible") ||
					road.getDescTerm().equals(
							"Private Road - Restricted Access")) {
						GlobalVariables.myPrint(
							"remove road element: " + road.getFID());
						this.getRoadContext().remove(road);
					}
				}
			}
		} catch (java.net.MalformedURLException e) {
			System.out.println("ContextCreator: malformed URL exception when reading
					roadshapefile. Check the 'roadLoc' parameter is correct");
			e.printStackTrace();
		}
	}

	public boolean onRoad(Coordinate c) {
		return false; //coordCache.containsKey(c);
	}

	public void BuildRoadNetwork() {
		if (VeinContextCreator.debug)
			GlobalVariables.myPrint("InfraContext: building road network");
		GeometryFactory geomFac = new GeometryFactory();
		int iLine = 0;
		Iterable<Object> roadElementIt
					= this.getRoadContext().getObjects(RoadElement.class);
		for (Object o : roadElementIt) {
			RoadElement road = (RoadElement) o;
			Geometry roadGeom = roadGeography.getGeometry(road);
			Coordinate c1 = roadGeom.getCoordinates()[0];
			Coordinate c2 = roadGeom.getCoordinates()[roadGeom.getNumPoints() - 1];
			Junction fromJunc = new Junction(c1, null);
			fromJunc.setName( "Junction" + fromJunc.getID());
```

```
Junction toJunc = new Junction(c2, null);
toJunc.setName( "Junction" + toJunc.getID());
boolean bPlus = false, bMinus = false;
if (existsInContext(fromJunc)) {
        fromJunc = getJunctionWithCoordinates(c1);
} else { // Junction does not exit
        this.getRoadContext().add(fromJunc);
        Point p1 = geomFac.createPoint(c1);
        roadGeography.move(fromJunc, p1);
}
if (existsInContext(toJunc)) {
        toJunc = getJunctionWithCoordinates(c2);
} else {
        this.getRoadContext().add(toJunc);
        Point p2 = geomFac.createPoint(c2);
        roadGeography.move(toJunc, p2);
}
if (road.getOneWay().equals("+")) {
        bPlus = true;
} else if (road.getOneWay().equals("-")) {
        bMinus = true;
} else if (road.getOneWay().equals("N")) {
        bPlus = true;
        bMinus = true;
} else {
        bPlus = true;
        bMinus = true;
}
road.setFromJunc(fromJunc);
road.setToJunc(toJunc);
fromJunc.addConnectedRoad(road);
toJunc.addConnectedRoad(road);
if (bPlus) {
        MyRepastEdge edge1 = new MyRepastEdge(
          fromJunc, toJunc, true, roadGeom.getLength(), road.getFID()+ "+");
        if (!roadNetwork.containsEdge(edge1)) {
                roadNetwork.addEdge(edge1);
        } else {
                System.err.println("InfraContext: buildRoadNetwork: for
                        some reason this edge that has just been created
                                already exists in the RoadNetwork!");
        }
}
if (bMinus) {
        MyRepastEdge edge2 = new MyRepastEdge(
           toJunc, fromJunc, true, roadGeom.getLength(), road.getFID()+ "-");
        if (!roadNetwork.containsEdge(edge2)) {
                roadNetwork.addEdge(edge2);
        } else {
                System.err.println("InfraContext: buildRoadNetwork: for
                        some reason this edge that has just been created
                                already exists in the RoadNetwork!");
        }
}
iLine++;
if ((iLine % 1000) == 1) GlobalVariables.myPrint(false, road.getFID() + " ");
if ((iLine % 10000) == 1) GlobalVariables.myPrint("");
```

```
		} // for road:
	}

	public RoadElement getRoadElementWithID(String id) {
		Iterable<Object> roadElementIt
					= this.getRoadContext().getObjects(RoadElement.class);
		for (Object o : roadElementIt) {
			RoadElement road = (RoadElement) o;
			if (road.getFID().equals(id)) {
				return road;
			}
		}
		System.err.println("InfraContext: getRoadElementWithID:
						Error, couldn't find a road woth id: " + id);
		return null;
	}

	public RoadEdge getRoadEdgeWithID(String id, boolean flip) {

		Iterable<Object> roadElementIt
					= this.getRoadContext().getObjects(RoadElement.class);
		for (Object o : roadElementIt) {
			RoadElement road = (RoadElement) o;
			if (road.getFID().equals(id)) {
				RoadEdge roadEdge = new RoadEdge (road);
				if (flip) roadEdge.setFlip(true);
				return roadEdge;
			}
		}
		System.err.println("InfraContext: getRoadEdgeWithID: Error,
				couldn't find a road woth id: "+ id);
		return null;
	}

	public int getNumOfEndingJunctions () {
		Iterable<Object> juncIt = this.getRoadContext().getObjects(Junction.class);
		int num = 0;

		for (Object o : juncIt) {
			Junction junc = (Junction) o;
			if (junc.getConnectedRoads().size() == 1) num++;
		}
		return num;
	}

	public int getNumOfJunctions (Junction ajunc, double distRange, boolean inside) {
		Iterable<Object> juncIt = this.getRoadContext().getObjects(Junction.class);
		int num = 0;
		double directDistance = 0;

		for (Object o : juncIt) {
			Junction myjunc = (Junction) o;
			directDistance = myjunc.getCoordinate().distance(ajunc.getCoordinate());
			if (inside && directDistance <= distRange) num++;
			else if (!inside && directDistance > distRange) num++;
		}
		return num;
	}

	public BrokenVehicle getABrokenVehicle(String name) {
		Iterable<Object> brokenVehicleIt
						= this.getRoadContext().getObjects(BrokenVehicle.class);
		for (Object o : brokenVehicleIt) {
			BrokenVehicle brokenVehicle = (BrokenVehicle) o;
			if (brokenVehicle.getVehicleType().equals(name)) { //"Ambulance")) {
				return brokenVehicle;
			}
		}
		return null;
	}

	public EmergencyVehicle getAnEmergencyVehicle(String name) {
		Iterable<Object> emergencyVehicleIt
					= this.getRoadContext().getObjects(EmergencyVehicle.class);
		for (Object o : emergencyVehicleIt) {
			EmergencyVehicle emergencyVehicle = (EmergencyVehicle) o;
			if (emergencyVehicle.getVehicleType().equals(name)) {
				return emergencyVehicle;
			}
		}
		return null;
	}

	public int getNumberOfPrivateVehicles(int preferredLane, boolean onlyDSRC) {
		Iterable<Object> pVehicleIt = this.getRoadContext().getObjects(PrivateVehicle.class);
		int num = 0;
		for (Object o : pVehicleIt) {
			PrivateVehicle pVehicle = (PrivateVehicle) o;
			if (pVehicle.getPreferedLane() == preferredLane) {
				if (onlyDSRC) {
					if (pVehicle.isBUseComm()) num++;
				} else num++;
			} else if (preferredLane == 0) {
				if (onlyDSRC) {
					if (pVehicle.isBUseComm()) num++;
				} else num++;
			}
		}
		return num;
	}

	public Context getRoadContext() {
		return this.maincontext.getSubContext(subcontextString);
	}

	public Geography getRoadGeography() {
		return (Geography) getRoadContext().getProjection(geographyString);
	} //"RoadGeography"

	public Network getRoadNetwork() {
		return (Network) getRoadContext().getProjection(Network.class, networkString);
	}
```

```
public Network getCommNetwork() {
        return (Network) getRoadContext().getProjection(Network.class, "CommNetwork");
}

public RoadElement findRoadAtCoordinates(Coordinate coord)
                                        throws NullPointerException {
        if (coord == null) {
                throw new NullPointerException(
               "InfraContext: findRoadAtCoordinates: ERROR: the input coordinate is null");
        }
        GeometryFactory geomFac = new GeometryFactory();

        // Use a buffer for efficiency
        Point point = geomFac.createPoint(coord);
        Geometry buffer = point.buffer(GlobalVariables.XXXX_BUFFER);
        double minDist = Double.MAX_VALUE;
        RoadElement nearestRoad = null;

        Iterable<Object> roadElementIt = roadGeography.getObjectsWithin(buffer
                        .getEnvelopeInternal(), RoadElement.class);
        for (Object o : roadElementIt) {
                RoadElement road = (RoadElement) o;
                DistanceOp distOp = new DistanceOp(point, roadGeography
                                .getGeometry(road));
                double thisDist = distOp.distance();
                if (thisDist < minDist) {
                        minDist = thisDist;
                        nearestRoad = road;
                } // if thisDist < minDist
        } // for nearRoads
        if (nearestRoad == null) {
                System.err.println("InfraContext: findRoadAtCoordinates: ERROR:
                        couldn't find a road at these coordinates:\n\t"+ coord.toString());
        }
        return nearestRoad;
}

public Junction getFixedDestJunc () {
        if (this.fixedDestJunc != null) return fixedDestJunc;

        int x1=0, y1=0;
        if (GlobalVariables.iCase == 1) {
                //Nearest junction from PrincessRoyalUniversityHospital
                x1 = 543585;
                y1 = 165031;
        } else if (GlobalVariables.iCase == 2) {
                //Nearest junction from University College Hospital
                x1 = 529417;
                y1 = 182374;
        } else if (GlobalVariables.iCase == 3) {
                //Nearest junction from Kingston Hospital
                x1 = 519651;
                y1 = 169856;
        }

        Coordinate coord = new Coordinate( x1, y1);
        Point point = new GeometryFactory().createPoint(coord);
```

```
		Geometry buffer = point.buffer(1.0);//GlobalVariables.XXXX_BUFFER);
		double minDist = Double.MAX_VALUE;
		Junction nearestJunc = null;

		Iterable<Object> junctionIt = this.roadGeography.getObjectsWithin(buffer
				.getEnvelopeInternal(), Junction.class);
		if (junctionIt == null) GlobalVariables.myPrint("getFixedDestJunc3-2: ");
		for (Object o : junctionIt) {
			Junction junc = (Junction) o;
			DistanceOp distOp
			= new DistanceOp(point, this.getRoadGeography().getGeometry(junc));
			double thisDist = distOp.distance();
			if (thisDist < minDist) {
				minDist = thisDist;
				nearestJunc = junc;
			} // if thisDist < minDist
		} // for nearRoads
		this.fixedDestJunc = nearestJunc;
		return this.fixedDestJunc;
	}

	public List<Object> findObjectsOnTheRoad(RoadElement currentRoad, Class type)
			throws NullPointerException {
		if (currentRoad == null) {
			throw new NullPointerException("InfraContext: findRoadAtCoordinates:
				ERROR: the input coordinate is null");
		}
		Coordinate[] roadCoords = roadGeography.getGeometry(currentRoad)
								.getCoordinates();
		GeometryFactory geomFac = new GeometryFactory();

		// Use a buffer for efficiency
		LineString line = geomFac.createLineString(roadCoords);
		Geometry buffer = line.buffer(GlobalVariables.XXXX_BUFFER);
		double minDist = Double.MAX_VALUE;
		RoadElement nearestRoad = null;

		List<Object> objectList = new ArrayList<Object>();
		Iterable<Object> objectIt = roadGeography.getObjectsWithin(line
				.getEnvelopeInternal(), type);
		for (Object o : objectIt) {
			GlobalVariables.myPrint(o.toString() + " is in the boundary of the road");
			Geometry geo = roadGeography.getGeometry(o);
			if (line.crosses(geo))
				GlobalVariables.myPrint(o.toString() + " crosses");
			if (line.covers(geo))
				GlobalVariables.myPrint(o.toString() + " covers");
			DecimalFormat df = new DecimalFormat("#.###############");
			Double dist1 = line.distance(geo);
			Double dist2 = 999.0;

			Coordinate oCoord = roadGeography.getGeometry(o).getCoordinate();
			RoadElement oRoad = this.findRoadAtCoordinates(oCoord);

			if (currentRoad.getFID().equals(oRoad.getFID())) {
				GlobalVariables.myPrint(o.toString() + " same road");
				objectList.add(o);
```

```
                        }
                }
                return objectList;
        }

        public static int random(int n, boolean bMath) {
                return random(1, n, bMath);
        }

        public static int random(int a, int b, boolean bMath) {
                if (bMath) return a + (int) (Math.random() * (b - a + 1));
                else return (int) RandomHelper.nextDoubleFromTo(a, b);
        }

        public int getNextVehicleNumber() {
                this.iVNum ++;
                return this.iVNum;
        }
}
```

D.2 Road geography and network

D.2.1 RoadElement.java

```
public class RoadElement {

        private String identifier;
        private String RoadLinkTO;
        private String desc;
        private String DescTerm;
        private String Nature;
        private String OneWay;
        private String NoEntry;
        private String HasTraffic; // this can be used when implementing traffic signals and controllers
        private double Shape_Leng;
        protected Junction fromJunc, toJunc;

        public RoadElement(String identifier, String desc) {
                this.identifier = identifier;
                this.desc = desc;
        }

        public RoadElement() {
                this.desc = "road";
        }

        public String toString() {
                return "road fid: " + identifier + ",  description: "+ desc + "-" + DescTerm;
        }

        public String toString2() {
                return identifier + ", "
                   + RoadLinkTO + ", "
                   + desc + ", "
                   + DescTerm + ", "
                   + OneWay + ", "
                   + NoEntry + ", "
                   + Shape_Leng;
        }

        public String getIdentifier()  {
                if (identifier.equals("") || identifier == null) {
                   System.err.println("Road: the identifier field for this road has not been initialised.");
                }
                return identifier;
        }

        public void setIdentifier(String identifier) {
                this.identifier = identifier;
        }

        public double getShape_Leng() {
                return Shape_Leng;
        }
```

```
public void setShape_Leng(double shape_Leng) {
	Shape_Leng = shape_Leng;
}

public String getFID() {
	return identifier;
}

public String getRoadLinkTO()  {
	if (RoadLinkTO.equals("") || RoadLinkTO == null) {
	   System.err.println("Road: the identifier field for this road has not been initialised."");
	}
	return RoadLinkTO;
}

public void setRoadLinkTO(String toid) {
	this.RoadLinkTO = toid;
}

public ArrayList<Junction> getJunctions() {
	ArrayList<Junction> junctions;	// The junctions at either end of the road
	junctions = new ArrayList<Junction>();
	junctions.add(fromJunc);
	junctions.add(toJunc);
	return junctions;
}

public Junction getFromJunc() {
	return fromJunc;
}

public void setFromJunc(Junction fromJunc) {
	this.fromJunc = fromJunc;
}

public Junction getToJunc() {
	return toJunc;
}

public void setToJunc(Junction toJunc) {
	this.toJunc = toJunc;
}

public String getOneWay() {
	return OneWay;
}

public void setOneWay(String oneWay) {
	OneWay = oneWay;
}

public String getNoEntry() {
	return NoEntry;
}

public void setNoEntry(String noEntry) {
```

```
                NoEntry = noEntry;
        }

        public String getDescTerm() {
                return DescTerm;
        }

        public void setDescTerm(String descTerm) {
                DescTerm = descTerm;
        }

        public String getNature() {
                return Nature;
        }

        public void setNature(String nature) {
                Nature = nature;
        }

        public String getHasTraffic() {
                return HasTraffic;
        }

        public void setHasTraffic(String hasTraffic) {
                HasTraffic = hasTraffic;
        }
}
```

D.2.2 RoadEdge.java

```
public class RoadEdge {

        private boolean flip = false;
        private RoadElement roadElement;

        public RoadElement getRoadElement() {
                return roadElement;
        }

        public void setRoad(RoadElement road) {
                this.roadElement = road;
        }

        public RoadEdge(RoadElement road) {
                this.roadElement = road;
        }

        public boolean isFlipped() {
                return flip;
        }

        public void setFlip(boolean flip) {
                this.flip = flip;
```

```
	}

	public String getName (){
		String direction;
		if (this.flip)direction = "-";
		else direction = "+";
		return this.roadElement.getFID() + direction;
	}

	public String getRoadElementName (){
		return this.roadElement.getFID();
	}

	public String getFullName () {
		return this.roadElement.fromJunc.getID() + "~" + this.getName() + "~"
									+ this.roadElement.toJunc.getID();
	}

	public boolean equals(RoadEdge r) {

		if (this.getName().equals(r.getName()))
			return true;
		else
			return false;
	}
}
```

D.2.3 MyRepastEdge.java

```
public class MyRepastEdge<T> extends RepastEdge<T> {

	private String edgeID;

	public MyRepastEdge(T source, T target, boolean directed, String edgeID) {
		this(source, target, directed, 1, edgeID);
	}

	public MyRepastEdge(T source, T target, boolean directed, double weight, String edgeID) {
		super (source, target, directed, weight);
		this.edgeID = edgeID;
	}

	public String getRoadID() {
		return edgeID.substring(0, edgeID.length()-1);
	}
	public String getEdgeID() {
		return edgeID;
	}
}
```

D.2.4 Junction.java

```
public class Junction {

        private int ID;
        protected String name;
        private Coordinate coord;
        // The Roads connected to this Junction
        private ArrayList<RoadElement> connectedRoads = new ArrayList<RoadElement>();

        public Junction() {
                this.ID = VeinContextCreator.generateAgentID();
        }

        public Junction(Coordinate coord, RoadElement road) {
                this.ID = VeinContextCreator.generateAgentID();
                this.coord = coord;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String toString() {
                return name + " at: " + this.coord.toString();
        }

        public int getID() {
                return ID;
        }
        public void setID(int id) {
                this.ID = id;
        }

        public Coordinate getCoordinate() {
                return this.coord;
        }

        public void setCoordinate(Coordinate coord) {
                this.coord = coord;
        }

        public boolean equals(Junction j) {
                if (this.coord.equals(j.getCoordinate())) {
                        if (this.getID() == j.getID()) return true;
                        else return false;
                }
                else
                        return false;
        }

        public ArrayList<RoadElement> getConnectedRoads() {
                return this.connectedRoads;
        }
        public void addConnectedRoad(RoadElement road) {
                this.connectedRoads.add(road);
        }

}
```

D.3 Vehicle and route

D.3.1 Vehicle.java

```
public class Vehicle {

        protected int id;
        private String name;
        private Junction originJunc;
        private Junction destJunc;
        protected boolean bSuddenDeceleration = false;

        protected double v0;
        protected double v;
        public static double T = 1.0;
        public static double S0 = 2.0;
        public static double delta = 4.0;
        public static double a = 1.0;
        public static double b = 1.5;
        public static double vlength = 5.0;
        public static double v0maxLane1 = 60.0;
        public static double v0maxLane2 = 70.0;
        public static double v0maxLane3 = 80.0;
        public static double conversionMph2Kmh = 1.609344;//mph to kmh
        public static double conversionMph2Ms = conversionMph2Kmh / 3.6;
        public static double conversionMs2Mph = 3.6 / conversionMph2Kmh;

        protected double speed;
        protected Route route;
        protected boolean bStopped;
        protected boolean bArrived;
        protected double dTravelTime;
        protected double dCountInfluence;
        protected Infra infra;
        protected CommunicationMessage veinOnt;
        protected boolean bRequested;
        protected double dStopInterval = 0;
        protected String vehicleType;
        private int preferedLane;

        public Vehicle(Infra veinInfra, String vehicleType, boolean bUseOnt) {
                this.infra = veinInfra;
                this.bStopped = false;
                this.bArrived = false;
                this.dTravelTime = 0.0;
```

```
		this.dCountInfluence = 0.0;
		this.dStopInterval = 0.0;
		this.bRequested = false;
		this.vehicleType = vehicleType;
		this.veinOnt = new CommunicationMessage();
	}

	public Junction getRandomJunctionFromRandomHelper () {
		Junction aJunc;
		int irandom = RandomHelper.nextIntFromTo(0,this.infra.getRoadContext()
									.getObjects(Junction.class).size() -1);
		aJunc = (Junction) this.infra.getRoadContext().getObjects(Junction.class).get(irandom);
		return aJunc;
	}

	public Junction getRandomJunction (boolean bMathRandom) {
		Junction aJunc;
		if (bMathRandom) {
			int irandom = Infra.random(0, this.infra.getRoadContext().
						getObjects(Junction.class).size() -1, bMathRandom);
			aJunc = (Junction) this.infra.getRoadContext().
						getObjects(Junction.class).get(irandom);
		} else aJunc = (Junction) this.infra.getRoadContext()
						.getRandomObjects(Junction.class, 1).iterator().next();
		return aJunc;
	}

	public Junction getRandomJunction () {
		boolean bMathRandom = false;
		Junction aJunc;
			if (bMathRandom) {
				int irandom = Infra.random(0, this.infra.getRoadContext()
						.getObjects(Junction.class).size() -1, bMathRandom);
				aJunc = (Junction) this.infra.getRoadContext()
						.getObjects(Junction.class).get(irandom);
			} else aJunc = (Junction) this.infra.getRoadContext()
						.getRandomObjects(Junction.class, 1).iterator().next();
		return aJunc;
	}

	public void initializeRoute(Junction originationJunc, Junction destinationJunc, Route refRoute){
		this.initializeRoute( originationJunc,  destinationJunc,  refRoute, 0 ) ;
	}

	public void initializeRoute(Junction originationJunc, Junction destinationJunc,
										Route refRoute, int nCase ) {
		double minimumRouteDistance;
		if (this.vehicleType.equals("PrivateVehicle") ||
			this.vehicleType.equals("PrivateVehicle2")) minimumRouteDistance = 1000.0;
		else minimumRouteDistance = GlobalVariables.MININUM_ROUTE_DISTANCE;
		if ( originationJunc == null &&  destinationJunc == null) {
			this.setOriginJuncRandomly();
			this.setDestJuncRandomly(this.originJunc, minimumRouteDistance);
		}
		else if (originationJunc != null && destinationJunc == null) {
			this.route = null;
			this.setDestJunc(null);
```

```
				this.setOriginJunc(originationJunc);
				this.setDestJuncRandomly(this.originJunc, minimumRouteDistance, nCase);
			}
			else if (originationJunc == null && destinationJunc != null) {
				this.route = null;
				this.setOriginJunc(null);
				this.setDestJunc(destinationJunc);
				this.setOriginJuncRandomly(this.destJunc, minimumRouteDistance, nCase);
			} else {
				this.route = null;
				this.setOriginJunc(originationJunc);
				this.setDestJunc(destinationJunc);
			}
			boolean bGetARoute = false;
			int iWhile = 0;
			do {
				bGetARoute = this.setRoute(this.originJunc, this.destJunc, refRoute);
				if (!bGetARoute) {
					if (originationJunc != null && destinationJunc == null) {
						if (iWhile < 10) {
							this.route = null;
							this.setOriginJuncRandomly(nCase);
							this.setDestJuncRandomly(this.originJunc,
										minimumRouteDistance, nCase);
						} else {
							this.setOriginJuncRandomly(nCase);
							this.setDestJunc(this.getOriginJunc());
							this.setOriginJunc(null);
							this.setOriginJuncRandomly(this.destJunc,
										minimumRouteDistance, nCase);
							iWhile = 0;
						}
					}
					else if (originationJunc == null && destinationJunc != null){
						this.setOriginJuncRandomly(this.destJunc,
										minimumRouteDistance, nCase);
					} else {
						this.setOriginJuncRandomly(nCase);
						this.setDestJuncRandomly(this.originJunc,
										minimumRouteDistance, nCase);
					}
				}
				iWhile ++;
			} while (!bGetARoute);
		}

		public void initializeRouteFromRandomHelper(Junction originationJunc,
								Junction destinationJunc, Route refRoute, int nCase ) {
			double minimumRouteDistance;
			if (this.vehicleType.equals("PrivateVehicle") ||
							this.vehicleType.equals("PrivateVehicle2")) {
				minimumRouteDistance = 1000.0;
			}
			else if (GlobalVariables.iCase == 4 ) {
				minimumRouteDistance = 1000.0;
			}
			else minimumRouteDistance = GlobalVariables.MININUM_ROUTE_DISTANCE;
```

```
if ( originationJunc == null &&  destinationJunc == null) {
        this.setOriginJuncRandomlyFromRandomHelper(nCase);
        this.setDestJuncRandomlyFromRandomHelper(this.originJunc,
                                        minimumRouteDistance, nCase);
}
else if (originationJunc != null && destinationJunc == null) {
        this.route = null;
        this.setDestJunc(null);
        this.setOriginJunc(originationJunc);
        this.setDestJuncRandomlyFromRandomHelper(this.originJunc,
                                        minimumRouteDistance, nCase);
}
else if (originationJunc == null && destinationJunc != null) {
        this.route = null;
        this.setOriginJunc(null);
        this.setDestJunc(destinationJunc);
        this.setOriginJuncRandomlyFromRandomHelper(this.destJunc,
                                        minimumRouteDistance, nCase);
} else {
        this.route = null;
        this.setOriginJunc(originationJunc);
        this.setDestJunc(destinationJunc);
}

boolean bGetARoute = false;
int iWhile = 0;

do {
        bGetARoute = this.setRoute(this.originJunc, this.destJunc, refRoute);
        if (!bGetARoute) {
                if (originationJunc != null && destinationJunc == null) {
                        if (iWhile < this.infra.getRoadContext()
                                        .getObjects(Junction.class).size()) {
                                this.route = null;
                                this.setDestJuncRandomlyFromRandomHelper(
                                  this.originJunc,  minimumRouteDistance, nCase);

                        } else {
                            this.setDestJuncRandomlyFromRandomHelper(nCase);
                            this.setOriginJuncRandomlyFromRandomHelper(
                                this.destJunc, minimumRouteDistance, nCase);
                            iWhile = 0;
                        }
                }
                else if (originationJunc == null && destinationJunc != null){
                        if (iWhile < this.infra.getRoadContext().
                                        getObjects(Junction.class).size()) {
                                this.route = null;
                                this.setOriginJuncRandomlyFromRandomHelper(
                                   this.destJunc, minimumRouteDistance, nCase);
                        } else {
                            this.setOriginJuncRandomlyFromRandomHelper(nCase);
                            this.setDestJuncRandomlyFromRandomHelper(
                                this.originJunc, minimumRouteDistance, nCase);
                            iWhile = 0;
                        }
                } else {
```

```
                                        if (iWhile <= this.infra.getNumOfEndingJunctions()) {
                                            this.setDestJuncRandomlyFromRandomHelper(nCase);
                                        } else {
                                            this.setOriginJuncRandomlyFromRandomHelper(nCase);
                                            this.setDestJuncRandomlyFromRandomHelper(
                                                    this.originJunc, minimumRouteDistance, nCase);
                                            iWhile = 0;
                                        }
                                }
                        }
                        iWhile ++;
                } while (!bGetARoute);
        }

        public void initializeRouteFromOriginFromRandomHelper(Junction originationJunc,
                                                        int nCase, double minimumRouteDistance ) {
           this.route = null;
                this.setDestJunc(null);
                if (originationJunc == null) {
                        this.setOriginJuncRandomlyFromRandomHelper(nCase);
                }
                this.setDestJuncRandomlyFromRandomHelper(this.originJunc,
                                                                minimumRouteDistance, nCase);

                boolean bGetARoute = false;
                int iWhile = 0;
                int numEndingJunctions = 0;
                numEndingJunctions = this.infra.getNumOfEndingJunctions();

                do {
                        double directDistance = this.getOriginJunc().getCoordinate()
                                                .distance(this.getDestJunc().getCoordinate());
                        bGetARoute = this.setRoute(this.originJunc, this.destJunc, null);
                        if (!bGetARoute) {
                                if (originationJunc != null ) {
                                        if (iWhile <= numEndingJunctions) {
                                                this.route = null;
                                                this.setDestJuncRandomlyFromRandomHelper(
                                                  this.originJunc, minimumRouteDistance, nCase);
                                        } else {
                                            this.setOriginJuncRandomlyFromRandomHelper(nCase);
                                            this.setDestJuncRandomlyFromRandomHelper(
                                                    this.originJunc, minimumRouteDistance, nCase);
                                            iWhile = 0;
                                        }
                                }
                        }
                        iWhile ++;
                } while (!bGetARoute);
        }
        public boolean updateRelativeLocation() {
                return this.veinOnt.updateRelativeLocation(this);
        }

        public String getVehicleType() {
                return vehicleType;
        }
```

```
public void setVehicleType(String vehicleType) {
	this.vehicleType = vehicleType;
}

public Junction getOriginJunc() {
	return this.originJunc;
}

public void setOriginJunc(Junction originJunc) {
	this.originJunc = originJunc;
}

public void setOriginJuncRandomly() {
	this.setOriginJuncRandomly(0);
}

public void setOriginJuncRandomly(int nCase) {
	Junction j = this.getRandomJunction();
	if (nCase == 4) {
		while (j.getConnectedRoads().size() != 1) {
			j = this.getRandomJunction();
		}
	}
	this.setOriginJunc(j);
}

public void setOriginJuncRandomlyFromRandomHelper(int nCase) {
	Junction j = this.getRandomJunctionFromRandomHelper();
	if (nCase == 4) {
		boolean bGetIt = false;
		do {
			if (j.getConnectedRoads().size() == 1) {
				if (j.equals(j.getConnectedRoads().get(0).getFromJunc())) {
					this.setOriginJunc(j);
					bGetIt = true;
				}
			}
			j = this.getRandomJunctionFromRandomHelper();
		} while (!bGetIt);
	}
	this.setOriginJunc(j);
}

public void setOriginJuncRandomly(Junction destJunc, double distance) {
	this.setOriginJuncRandomly(destJunc, distance, 0);
}

public void setOriginJuncRandomly(Junction destJunc, double distance, int nCase) {
	Junction aJunc;
	Coordinate originCoord, destCoord;
	do {
		aJunc = this.getRandomJunction();
		if (nCase == 4) {
			while (aJunc.getConnectedRoads().size() != 1) {
				aJunc = this.getRandomJunction();
			}
			}
			if (nCase == 41) {
				boolean isJuncConnectedToDualCarriageway = false;
				while (!isJuncConnectedToDualCarriageway) {
					Iterator<RoadElement> i;
					i = aJunc.getConnectedRoads().iterator();
					while (i.hasNext()) {
						RoadElement aRoad = (RoadElement) i.next();
						if(aRoad.getNature().equals("Dual Carriageway")){
						    isJuncConnectedToDualCarriageway = true;
						    break;
						}
					}
					if (!isJuncConnectedToDualCarriageway)
							aJunc = this.getRandomJunction();
				}
			}
			originCoord = aJunc.getCoordinate();
			destCoord = destJunc.getCoordinate();
		} while (originCoord.distance(destCoord)== 0 ||
				originCoord.distance(destCoord) < distance);

		if (this.originJunc != null) {
			if (this.originJunc.equals(aJunc)) {
				do {
					aJunc = this.getRandomJunction(true);
					if (nCase == 4) {
						while (aJunc.getConnectedRoads().size() != 1) {
							aJunc = this.getRandomJunction();
						}
					}
					if (nCase == 41) {
					    boolean isJuncConnectedToDualCarriageway = false;
					    while (!isJuncConnectedToDualCarriageway) {
						Iterator<RoadElement> i;
						i = aJunc.getConnectedRoads().iterator();
						while (i.hasNext()) {
						RoadElement aRoad = (RoadElement) i.next();
						if(aRoad.getNature().equals("Dual Carriageway")){
						    isJuncConnectedToDualCarriageway = true;
						    break;
						 }
						}
						if (!isJuncConnectedToDualCarriageway)
							aJunc = this.getRandomJunction();
						}
					}
					originCoord = aJunc.getCoordinate();
					destCoord = destJunc.getCoordinate();

				} while (originCoord.distance(destCoord)== 0 ||
					originCoord.distance(destCoord) < distance);
			}
		}
		this.originJunc = aJunc;
	}
```

```
public void setOriginJuncRandomlyFromRandomHelper(Junction destJunc,
                                                  double distance, int nCase) {
        Junction aJunc;
        int iwhile = 0;
        Coordinate originCoord, destCoord;
        do {
                aJunc = this.getRandomJunctionFromRandomHelper();
                if (nCase == 4) {
                        boolean bGetIt = false;
                        do {
                                if (aJunc.getConnectedRoads().size() == 1) {
                                        if (aJunc.equals(aJunc.getConnectedRoads()
                                                        .get(0).getFromJunc())) {
                                                this.setOriginJunc(aJunc);
                                                bGetIt = true;
                                        }
                                }
                                aJunc = this.getRandomJunctionFromRandomHelper();
                        } while (!bGetIt);
                }
                originCoord = aJunc.getCoordinate();
                destCoord = destJunc.getCoordinate();
                iwhile++;
        } while (originCoord.distance(destCoord)== 0
                        || originCoord.distance(destCoord) < distance);
        if (this.originJunc != null) {
                if (this.originJunc.equals(aJunc)) {
                        do {
                         aJunc = this.getRandomJunctionFromRandomHelper();
                           if (nCase == 4) {
                             boolean bGetIt = false;
                             do {
                               if (aJunc.getConnectedRoads().size() == 1) {
                                 if (aJunc.equals(aJunc.getConnectedRoads()
                                                        .get(0).getFromJunc())) {
                                      this.setOriginJunc(aJunc);
                                      bGetIt = true;
                                 }
                               }
                               aJunc = this.getRandomJunctionFromRandomHelper();
                             } while (!bGetIt);
                           }
                          originCoord = aJunc.getCoordinate();
                          destCoord = destJunc.getCoordinate();
                        } while (originCoord.distance(destCoord)== 0 ||
                                        originCoord.distance(destCoord) < distance);
                }
        }
        this.originJunc = aJunc;
}

public Junction getDestJunc() {
        return destJunc;
}

public void setDestJunc(Junction destJunc) {
        this.destJunc = destJunc;
```

```
}

public void setDestJuncRandomly(int nCase) {
        Junction j = this.getRandomJunction();
        if (nCase == 4) {
                while (j.getConnectedRoads().size() != 1) {
                        j = this.getRandomJunction();
                }
        }
        this.setDestJunc(j);
}

public void setDestJuncRandomlyFromRandomHelper(int nCase) {
        Junction j = this.getRandomJunctionFromRandomHelper();
        if (nCase == 4) {
                boolean bGetIt = false;
                do {
                        if (j.getConnectedRoads().size() == 1) {
                                if (j.equals(j.getConnectedRoads().get(0).getToJunc())) {
                                        this.setDestJunc(j);
                                        bGetIt = true;
                                }
                        }
                        j = this.getRandomJunctionFromRandomHelper();
                } while (!bGetIt);
        }
        this.setDestJunc(j);
}

public void setDestJuncRandomly(Junction originJunc, double distance) {
        this.setDestJuncRandomly(originJunc, distance, 0);
}

public void setDestJuncRandomlyFromRandomHelper(Junction originJunc, double distance,
                                                                        int nCase) {
        Junction aJunc;
        int iwhile = 0;
        Coordinate originCoord, destCoord;
        do {
                aJunc = this.getRandomJunctionFromRandomHelper();
                if (nCase == 4) {
                        boolean bGetIt = false;
                        do {
                                if (aJunc.getConnectedRoads().size() == 1) {
                                        if (aJunc.equals(aJunc.getConnectedRoads()
                                                        .get(0).getToJunc())) {
                                                this.setDestJunc(aJunc);
                                                bGetIt = true;
                                        }
                                }
                                aJunc = this.getRandomJunctionFromRandomHelper();
                        } while (!bGetIt);
                }
                destCoord = aJunc.getCoordinate();
                originCoord = originJunc.getCoordinate();
                iwhile++;
        } while (originCoord.distance(destCoord)== 0
                    || originCoord.distance(destCoord) < distance);
        if (this.destJunc != null) {
                if (this.destJunc.equals(aJunc)) {
                        do {
                                aJunc = this.getRandomJunctionFromRandomHelper();
                                if (nCase == 4) {
                                        boolean bGetIt = false;
                                        do {
                                          if (aJunc.getConnectedRoads().size() == 1) {
                                            if (aJunc.equals(aJunc.getConnectedRoads()
                                                                .get(0).getToJunc())) {
                                              this.setDestJunc(aJunc);
                                              bGetIt = true;
                                            }
                                          }
                                     aJunc = this.getRandomJunctionFromRandomHelper();
                                        } while (!bGetIt);
                                }
                                destCoord = aJunc.getCoordinate();
                                originCoord = originJunc.getCoordinate();

                        } while (originCoord.distance(destCoord)== 0 ||
                                originCoord.distance(destCoord) < distance);
                }
        }
        this.destJunc = aJunc;
}

public void setDestJuncRandomly(Junction originJunc, double distance, int nCase) {
        Junction aJunc;
        Coordinate originCoord, destCoord;
        do {
                aJunc = this.getRandomJunction();
                if (nCase == 4) {
                        while (aJunc.getConnectedRoads().size() != 1) {
                                aJunc = this.getRandomJunction();
                        }
                }
                destCoord = aJunc.getCoordinate();
                originCoord = originJunc.getCoordinate();

        } while (originCoord.distance(destCoord)== 0
                                || originCoord.distance(destCoord) < distance);

        if (this.destJunc != null) {
                if (this.destJunc.equals(aJunc)) {
                        do {
                                aJunc = this.getRandomJunction(true);
                                if (nCase == 4) {
                                        while (aJunc.getConnectedRoads().size() != 1) {
                                                aJunc = this.getRandomJunction(true);
                                        }
                                }
                                destCoord = aJunc.getCoordinate();
                                originCoord = originJunc.getCoordinate();
                        } while (originCoord.distance(destCoord)== 0
                                        || originCoord.distance(destCoord) < distance);
```

```
			}
		}
		this.destJunc = aJunc;
	}

	public boolean setRoute (Junction originJunc, Junction destJunc, Route refRoute) {
		boolean bCheck = false;
		boolean bRefCheck = false;
		if (originJunc.equals(destJunc)) {
			return false;
		}
		try {
			if (this.route != null) {
				this.route.kill();
			}
			this.route = new Route(infra, this, originJunc, destJunc);
			bCheck = this.route.checkRouteCoords();
			if(refRoute != null) {
				for (int i =0; i < this.route.getRouteJunctions().size()-1; i++) {
					if (refRoute.getRouteJunctions()
						.contains(this.route.getRouteJunctions().get(i))) {
						bRefCheck = true;
						break;
					}
				}
				if (!bRefCheck) {
					this.route = null;
					return bRefCheck;
				}
			}
			if (!bCheck) this.route = null;
			return bCheck;
		} catch (Exception e) {
			e.printStackTrace();
			RunEnvironment.getInstance().pauseRun();
		}
		return false;
	}

	public boolean isStopped() {
		if (bStopped ) return true;
		else return false;
	}

	public void setStopped(boolean stopped) {
		this.bStopped = stopped;
		this.veinOnt.setStopped(stopped);
	}

	public void stop() {
		this.setStopped(true);
	}

	public boolean isArrived() {
		return bArrived;
	}
```

```
public void moveOverToTheRightLane() {
	this.setPreferedLane(2);
	this.veinOnt.setPreferedLane(2);
	if (this.v0 < this.v0maxLane2) {
		this.v0 = this.v0maxLane2; //(int) (this.v0 + this.v0maxLane2) / 2;
		double s = v0;
		if (this.speed == 0) s = 0;
		this.bSuddenDeceleration = false;
		this.setSpeed(s);
		this.veinOnt.setSpeed(s);
	}
}

public void die(){
	if (infra.getRoadContext().size() > 1)
		infra.getRoadContext().remove(this);
	else
		RunEnvironment.getInstance().endRun();
}

public void removeFromTheContext() {
	try {
	// Get the context in which the agent resides.
	Context context = this.infra.getRoadContext();
	//	Remove the agent from the context if the context is not empty
	if (context.size() > 1)
		context.remove(this);
	// Otherwise if the context is empty, end the simulation
	else
		RunEnvironment.getInstance().endRun();
	} catch (Exception e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
}

public void setArrived(boolean arrived) {
	bArrived = arrived;
	GlobalVariables.myPrint(this.toString() +  " is arrived at its destination");
}

public double getDTravelTime() {
	return dTravelTime;
}

public void setDTravelTime(double travelTime) {
	dTravelTime = travelTime;
}

public void addTravelTime(double add) {
	dTravelTime = dTravelTime + add;
}

public void addStopInterval (double add) {
	dStopInterval = dStopInterval + add;
	PrivateVehicle.totalStoppedInterval = PrivateVehicle.totalStoppedInterval + add;
}
```

```
public void setStopInterval (double time) {
        dStopInterval = time;
}

public void resetStopVariables () {
        dStopInterval = 0.0;
}

public double getStopInterval () {
        return dStopInterval;
}

public double getDCountInfluence() {
        return dCountInfluence;
}

public void setDInfluenceTime(double influenceTime) {
        dCountInfluence = influenceTime;
}

public void addInfluenceTime(double add) {
        dCountInfluence = dCountInfluence + add;
        PrivateVehicle.totalCountInfluence = PrivateVehicle.totalCountInfluence + add;
}

public double getSpeed() {
        return speed;
}

public void setSpeed(double speed) {
        this.speed = speed;
}

public void setSpeedAndPreferedLane(int nRandom) {
        double speed = 0;
        if (GlobalVariables.iCase < 4)  {
        /*
        Under 20 mph      5:5
        20-29 mph                49:54
        30-34 mph                30:84
        35-39 mph                12:96
        40-44 mph                3:99
        45-49 mph                1:100
        50 mph and over          0
        ----------------------------
        Percentage over 35 mph   16
        Average speed (mph)      30
        */

        speed = 30.0;
        if (nRandom <= 5) speed = 20;
        else if (nRandom > 5 && nRandom <= 54 ) speed = this.infra.random (20, 29, false);
        else if (nRandom > 54 && nRandom <= 84 ) speed = this.infra.random (30, 34, false);
        else if (nRandom > 84 && nRandom <= 96 ) speed = this.infra.random (35, 39, false);
        else if (nRandom > 96 && nRandom <= 99 ) speed = this.infra.random (40, 44, false);
        else if (nRandom == 100) speed = this.infra.random (45, 49, false);
```

```
		this.setSpeed(speed);
		this.setPreferedLane(1);
		} else if (GlobalVariables.iCase == 4) {
		/*
		Under 50 mph	4 % :04
		50-59 mph		14 %:18
		60-64 mph		14 %:32
		65-69 mph		19 %:51
		70-74 mph		21 %:72
		75-79 mph		15 %:87
		80-89 mph		12 %:99
		90 mph and over  2 % -> 1% :100
		--------------------------------
		More than 10 mph over limit	14 %
		Average speed (mph)	69 mph
		*/
		speed = 69.0;
		if (nRandom <= 4) speed = this.infra.random (49, 50, false);
		else if (nRandom > 4 && nRandom <= 18 ) speed = this.infra.random (50, 59, false);
		else if (nRandom > 18 && nRandom <= 32 ) speed = this.infra.random (60, 64, false);
		else if (nRandom > 32 && nRandom <= 51 ) speed = this.infra.random (65, 69, false);
		else if (nRandom > 51 && nRandom <= 72 ) speed = this.infra.random (70, 74, false);
		else if (nRandom > 72 && nRandom <= 87 ) speed = this.infra.random (75, 79, false);
		else if (nRandom > 87 && nRandom <= 99 ) speed = this.infra.random (80, 89, false);
		else if (nRandom == 100) speed = this.infra.random (90, 91, false);

		//In the simulation, it is regarded that Motorways have three lanes
		//the left-hand lane, the middle lane and the outer lane.
		//lane number 1, 2, 3 represents the left-hand lane, the middle lane, and the outer lane
		this.setSpeed(speed);
		if(speed < 65) this.setPreferedLane(1); //18% + 14 = 32 %
		else if(speed < 75) this.setPreferedLane(2); //54% - 14% = 40%
		else this.setPreferedLane(3); //29%-1% = 28%
		}
		this.v0 = speed;
		this.v = speed;
	}

	public int getPreferedLane() {
		return preferedLane;
	}

	public void setPreferedLane(int preferedLane) {
		this.preferedLane = preferedLane;
	}

	public void accerlate () {
		accerlate (10.0);
	}

	public void accerlate (double a) {
		setSpeed(getSpeed() + a);
		if (this.speed > 0) this.bStopped = false;
	}

	public void suddenDeceleration () {
		this.bSuddenDeceleration = true;
		this.decelerate(5);
	}

	public void decelerationIDM (double speed1, double s1, double timeUnit1) {
		double v1 = speed1 * Vehicle.conversionMph2Ms;
		double deacc1 = Math.pow((v1*v1)/(2.0*s1), 2) / this.b;
		this.decelerate (deacc1 * Vehicle.conversionMs2Mph * timeUnit1); // /10 or not
	}

	public void decelerate () {
		this.decelerate(5);
	}

	public void decelerate (double d) {
		if(this.speed > d) this.speed = this.speed -d;
		else {
			this.speed = 0;
		}
	}

	public boolean equals(Vehicle v) {
		if (this.name.equals(v.name))
			return true;
		else
			return false;
	}

	public Coordinate initialstep() {
		double totalLength = this.route.getTotalLenghtOfTheRoute();
		double travelPerTurn = this.getSpeed()
		  * GlobalVariables.TRAVEL_PER_TURN/GlobalVariables.TRAVEL_SPEED_MPH;
		travelPerTurn = travelPerTurn / 10.0;  // 1 tick = 0.1 sec
		int totalTick = (int) (totalLength / travelPerTurn);
		int min = (int) (totalTick * 0.01);
		int max = (int) (totalTick * 0.99);
		int loop = this.infra.random(min, max, false);

		if (this.getVehicleType().equals("BrokenCar")) {
			loop = (int) (totalTick / 2);
			for (int i = 0; i < loop; i++) this.route.travelA(true);
		} else if (this.getVehicleType().equals("PrivateVehicle")) {
			for (int i = 0; i < loop; i++) this.route.travelA(true);
		}
		return this.getCurrentCoordinate();
	}

	public void step() {
		double time = System.currentTimeMillis();
		try {
			if (this.route == null)  {
			}
			else {
				if (this.route.atDestination() && !this.isStopped()) {
					this.setStopped(true);
				}
			}
		}
		catch (Exception e) {
				e.printStackTrace();
				RunEnvironment.getInstance().pauseRun();
		}
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public String toString() {
		return name + ", " + this.dTravelTime + ": " ;
	}

	public void setName(String name) {
		this.name = name;
	}

	public Geometry getCurrentGeometry () {
		return infra.getRoadGeography().getGeometry(this);
	}

	public Coordinate getCurrentCoordinate () {
		return infra.getRoadGeography().getGeometry(this).getCoordinate();
	}

	public String getCurrentCoorinatesString () {
		double x = infra.getRoadGeography().getGeometry(this).getCoordinate().x;
		double y = infra.getRoadGeography().getGeometry(this).getCoordinate().y;
		return x + "," + y;
	}

	public double getCurrentCoordinateX () {
		return infra.getRoadGeography().getGeometry(this).getCoordinate().x;
	}

	public double getCurrentCoordinateY () {
		return infra.getRoadGeography().getGeometry(this).getCoordinate().y;
	}

	public String getCurrentRoadFID () {

		if (this.veinOnt !=null && this.veinOnt.getCurrentRoadEdge() != null)
			return this.veinOnt.getCurrentRoadEdge().getRoadElement().getFID();
		else return "-1";
	}

	public int getNextJunctionID () {
        if (this.veinOnt !=null && this.veinOnt.getNextJunction() != null)
            return this.veinOnt.getNextJunction().getID();
        else return -1;
    }

    public double getRemainDistanceToNextJunction () {
        if (this.veinOnt != null)
            return this.veinOnt.getRemainingDistanceToNextJunction();
        else return -1.0;
    }

    public RoadElement getCurrentRoadOn() {
        Coordinate currentCoord
            = infra.getRoadGeography().getGeometry(this).getCoordinate();
        return  infra.findRoadAtCoordinates(currentCoord);
    }

    public <X> Iterable<X> getObjects (Envelope env, Class<X> type) {
        return (Iterable<X>) infra.getRoadGeography().getObjectsWithin(env, type);
    }

    public boolean getRequest() {
        return true;
    }

    public boolean answerback() {
        return true;
    }

    public CommunicationMessage getVeinOnt() {
        return veinOnt;
    }

    public static boolean IsInsideTheCircle (double x, double y, double centre_x, double centre_y,
                                                                    double radius) {
        //including on the circle
        if (Math.pow(x - centre_x, 2) + Math.pow(y - centre_y, 2) <= Math.pow(radius, 2))
            return true;
        //only inside of the circle
        else return false;
    }

    public static double getDistance (double x, double y, double centre_x, double centre_y) {
        double distance = Math.pow(x - centre_x, 2) + Math.pow(y - centre_y, 2);
        distance = Math.pow(distance, 0.5);
        return distance;
    }
}
```

D.3.2 EmergencyVehicle.java

```
public class EmergencyVehicle extends Vehicle {
```

```
private int numRequest;
private int numRequestOut;
private boolean bBroken;
private int endTick;
private ArrayList<String> lstDSRCVehicleRequested;
private ArrayList<String> lstNonDSRCVehicleRequested;

public EmergencyVehicle(Infra veinInfra, String vehicleType, boolean bUseOnt) {
		super(veinInfra, vehicleType, bUseOnt);
		numRequest = 0;
		numRequestOut = 0;
		this.lstDSRCVehicleRequested = null;
		this.lstNonDSRCVehicleRequested = null;
		this.lstDSRCVehicleRequested = new ArrayList<String>();
		this.lstNonDSRCVehicleRequested = new ArrayList<String>();
}

public String getEVOutputter() {
		return this.getName()+ ","
		+ this.getSpeed() + ","
		+ this.infra.formatStr( this.getCurrentCoordinateX()) + ","
		+ this.infra.formatStr( this.getCurrentCoordinateY()) + ","
		+ this.getCurrentRoadFID() + ","
		+ this.getNextJunctionID() + ","
		+ this.infra.formatStr( this.getRemainDistanceToNextJunction()) + ","
		+ this.getNumRequest() + ","
		+ this.getNumRequestOut() + ","
		+ this.getNumStoppedPrivateVehiclePerTick() + ","
		+ this.getDTravelTime()+ ","
		+ this.isArrived()+ ","
		+ GlobalVariables.iRondomSeed+ ","
		+ GlobalVariables.iPercentUsingComm + ","
		+ GlobalVariables.numVehicle + ","
		+ GlobalVariables.numEmergencyVehicle + ","
		+ GlobalVariables.commrange + ","
		+ this.getNumOfVehicleRequested() + ","
		+ this.getNumOfDSRCVehicleRequested() + ","
		+ this.getNumOfNonDSRCVehicleRequested() + ","
		 + this.getNumPassedByPrivateVehicle() + ","
		+ this.getNumPassedByWithDSRCPrivateVehicle() + ","
		+ this.getNumPassedByWithoutDSRCPrivateVehicle() + ","
		+ this.getNumOfVehicleStopped() + ","
		+ this.getNumOfDSRCVehicleStopped() + ","
		+ this.getNumOfNonDSRCVehicleStopped() + ","
		+ this.getNumOfVehiclePassedBy() + ","
		+ this.getNumOfDSRCVehiclePassedBy() + ","
		+ this.getNumOfNonDSRCVehiclePassedBy() + ","
		+ PrivateVehicle.totalNumStopped + ","
		+ PrivateVehicle.totalStoppedInterval  + ","
		+ PrivateVehicle.totalCountInfluence + ","
		+ this.getCurrentCoorinatesString() + ","
		+ "||," + this.infra.getNumberOfPrivateVehicles(1, true)  + "," //about 32% * ont%
		+ this.infra.getNumberOfPrivateVehicles(1, false)  + "," //about 32%
		+ this.infra.getNumberOfPrivateVehicles(0, true) + "," // ont %
		+ GlobalVariables.numVehicle +", "
		+  GlobalVariables.totalNetworkLength + ","
		+ "ori," + this.getOriginJunc().getCoordinate().x + ","+
				this.getOriginJunc().getCoordinate().y + ","
			+ "des," + this.getDestJunc().getCoordinate().x + ","
				+ this.getDestJunc().getCoordinate().y;
	}

	public String getEVOutputter2() {
		return this.getName()+ ","
		+ this.getSpeed() + ","
		+ this.getDTravelTime()+ ","
		+ this.isArrived()+ ","
		+ GlobalVariables.iRondomSeed+ ","
		+ GlobalVariables.iPercentUsingComm + ","
		+ GlobalVariables.numVehicle + ","
		+ GlobalVariables.commrange + ","
		+ this.getNumOfVehicleRequested() + ","
		+ this.getNumOfDSRCVehicleRequested() + ","
		+ this.getNumOfNonDSRCVehicleRequested() + ","
		+ this.getNumOfVehicleStopped() + ","
		+ this.getNumOfDSRCVehicleStopped() + ","
		+ this.getNumOfNonDSRCVehicleStopped();
	}

	@Override
	public void step() {
		try {
		GlobalVariables.GarbageCollector("!!" + this.toString(), true);
		if (this.veinOnt.getCurrentRoadEdge() == null)
			this.updateRelativeLocation();
		if (!this.isStopped() || !this.isArrived()) {
			this.request();
		}
		if (this.dTravelTime >= this.endTick) {
			RunEnvironment.getInstance().endRun();
		}
		if (!this.isArrived()) {
			if (this.route.atDestination()) {
				this.setArrived(true);
				this.route = null;
				this.setStopped(true);
				RunEnvironment.getInstance().endRun();
			}
		}
		if (this.route != null && !this.isStopped()) {
			if (!this.bBroken) this.route.travelA(true);
			this.addTravelTime(1.0);
			this.updateRelativeLocation();
		}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
	public void request() {
		if (GlobalVariables.iCase < 4) requestOfAmbulance();
	}

	public void requestOfAmbulance() {
		this.numRequest = 0;
		this.numRequestOut = 0;
		PrivateVehicle.resetTotalNumStoppedPerTick();
		double a_x, a_y, centre_x, centre_y;
		centre_x = this.getCurrentCoordinateX();
		centre_y = this.getCurrentCoordinateY();

		Envelope env  = (Envelope)this.getCurrentGeometry().buffer
				(GlobalVariables.commrange * 1.5).getEnvelopeInternal();
		Iterable<PrivateVehicle> vehicleIt = this.getObjects(env, PrivateVehicle.class);
		for (PrivateVehicle v:vehicleIt) {
			if (PrivateVehicle.inTheVehiclePassedByList(v)) {
				continue;
			}
			a_x = v.getCurrentCoordinateX();
			a_y = v.getCurrentCoordinateY();
			double distance = getDistance(a_x, a_y, centre_x, centre_y);
			if (distance <= GlobalVariables.commrange) {
				v.getRequest(this.getVeinOnt(), this.getCurrentCoordinate());
				this.numRequest ++;
				addRequestedVehicleToTheList(v);
			}
			else if (v.isStopped()) {
				v.getRequest(this.getVeinOnt(), this.getCurrentCoordinate());
				this.numRequestOut ++;
			}
		}
	}

	public int getNumRequest() {
		return numRequest;
	}

	public int getNumRequestOut() {
		return numRequestOut;
	}

	public int getNumStoppedPrivateVehiclePerTick() {
		return PrivateVehicle.getTotalNumStoppedPerTick();
	}

	public int getNumPassedByPrivateVehicle () {
		return PrivateVehicle.getTotalNumPassedBy();
	}

	public int getNumPassedByWithDSRCPrivateVehicle () {
		return PrivateVehicle.getTotalNumPassedByWithDSRC();
	}

	public int getNumPassedByWithoutDSRCPrivateVehicle () {
		return PrivateVehicle.getTotalNumPassedByWithoutDSRC();
	}

	public int getNumOfVehicleStopped() {
		return  PrivateVehicle.getNumOfVehicleStopped();
	}

	public int getNumOfDSRCVehicleStopped() {
		return PrivateVehicle.getNumOfDSRCVehicleStopped();
	}
	public int getNumOfNonDSRCVehicleStopped() {
		return PrivateVehicle.getNumOfNonDSRCVehicleStopped();
	}

	public int getNumOfVehiclePassedBy() {
		return PrivateVehicle.getNumOfVehiclePassedBy();
	}

	public int getNumOfDSRCVehiclePassedBy() {
		return PrivateVehicle.getNumOfDSRCVehiclePassedBy();
	}
	public int getNumOfNonDSRCVehiclePassedBy() {
		return PrivateVehicle.getNumOfNonDSRCVehiclePassedBy();
	}

	public int getNumOfVehicleRequested() {
		return lstDSRCVehicleRequested.size() + lstNonDSRCVehicleRequested.size();
	}

	public int getNumOfDSRCVehicleRequested() {
		return lstDSRCVehicleRequested.size();
	}

	public int getNumOfNonDSRCVehicleRequested() {
		return lstNonDSRCVehicleRequested.size();
	}

	public void addRequestedVehicleToTheList(PrivateVehicle pv) {
		String vehicleName = pv.getName();
		if (pv.isBUseComm()) {
			if (!lstDSRCVehicleRequested.contains(vehicleName))
				lstDSRCVehicleRequested.add(vehicleName);
		} else {
			if (!lstNonDSRCVehicleRequested.contains(vehicleName))
				lstNonDSRCVehicleRequested.add(vehicleName);
		}
	}

	public boolean isBroken() {
		return bBroken;
	}

	public void setBroken(boolean broken) {
		this.bBroken = broken;
	}

	public int getEndTick() {
		return endTick;
	}

	public void setEndTick(int endTick) {
		this.endTick = endTick;
	}
}
```

D.3.3 BrokenVehicle.java

```
public class BrokenVehicle extends Vehicle {

        private int numRequest;
        private int numRequestOut;
        private boolean bBroken;
        private int endTick;
        private ArrayList<String> lstDSRCVehicleRequested;
        private ArrayList<String> lstNonDSRCVehicleRequested;

        public BrokenVehicle(Infra veinInfra, String vehicleType, boolean bUseOnt) {
                super(veinInfra, vehicleType, bUseOnt);
                numRequest = 0;
                numRequestOut = 0;
                this.lstDSRCVehicleRequested = null;
                this.lstNonDSRCVehicleRequested = null;
                this.lstDSRCVehicleRequested = new ArrayList<String>();
                this.lstNonDSRCVehicleRequested = new ArrayList<String>();
        }

        public String getEVOutputter() {
                return this.getName()+ ","
                 + this.getSpeed() + ","
                 + this.infra.formatStr( this.getCurrentCoordinateX()) + ","
                 + this.infra.formatStr( this.getCurrentCoordinateY()) + ","
                 + this.getCurrentRoadFID() + ","
                 + this.getNextJunctionID() + ","
                 + this.infra.formatStr( this.getRemainDistanceToNextJunction()) + ","
                 + this.getNumRequest() + ","
                 + this.getNumRequestOut() + ","
                 + this.getNumStoppedPrivateVehiclePerTick() + ","
                 + this.getDTravelTime()+ ","
                 + this.isArrived()+ ","
                 + GlobalVariables.iRondomSeed+ ","
                 + GlobalVariables.iPercentUsingComm + ","
                 + GlobalVariables.numVehicle + ","
                 + GlobalVariables.numEmergencyVehicle + ","
                 + GlobalVariables.commrange + ","
                 + this.getNumOfVehicleRequested() + ","
                 + this.getNumOfDSRCVehicleRequested() + ","
                 + this.getNumOfNonDSRCVehicleRequested() + ","
                 + this.getNumPassedByPrivateVehicle() + ","
                 + this.getNumPassedByWithDSRCPrivateVehicle() + ","
                 + this.getNumPassedByWithoutDSRCPrivateVehicle() + ","
                 + this.getNumOfVehicleStopped() + ","
                 + this.getNumOfDSRCVehicleStopped() + ","
                 + this.getNumOfNonDSRCVehicleStopped() + ","
                 + this.getNumOfVehiclePassedBy() + ","
                 + this.getNumOfDSRCVehiclePassedBy() + ","
                 + this.getNumOfNonDSRCVehiclePassedBy() + ","
                 + PrivateVehicle.totalNumStopped + ","
                 + PrivateVehicle.totalStoppedInterval  + ","
                 + PrivateVehicle.totalCountInfluence + ","
                 + this.getCurrentCoorinatesString() + ","
```

```
            + "||," + this.infra.getNumberOfPrivateVehicles(1, true)  + "," //about 32% * ont%
            + this.infra.getNumberOfPrivateVehicles(1, false)  + "," //about 32%
            + this.infra.getNumberOfPrivateVehicles(0, true) + "," // ont %
            + GlobalVariables.numVehicle +", "
            +  GlobalVariables.totalNetworkLength + ","
            + "ori," + this.getOriginJunc().getCoordinate().x + ","
                    + this.getOriginJunc().getCoordinate().y + ","
            + "des," + this.getDestJunc().getCoordinate().x + ","
                    + this.getDestJunc().getCoordinate().y;
}

public String getEVOutputter2() {
        return this.getName()+ ","
         + this.getSpeed() + ","
         + this.getDTravelTime()+ ","
         + this.isArrived()+ ","
         + GlobalVariables.iRondomSeed+ ","
         + GlobalVariables.iPercentUsingComm + ","
         + GlobalVariables.numVehicle + ","
         + GlobalVariables.commrange + ","
         + this.getNumOfVehicleRequested() + ","
         + this.getNumOfDSRCVehicleRequested() + ","
         + this.getNumOfNonDSRCVehicleRequested() + ","
         + this.getNumOfVehicleStopped() + ","
         + this.getNumOfDSRCVehicleStopped() + ","
         + this.getNumOfNonDSRCVehicleStopped();
}

@Override
public void step() {
   try {
        GlobalVariables.GarbageCollector("!!" + this.toString(), true);
        if (this.veinOnt.getCurrentRoadEdge() == null)
                this.updateRelativeLocation();
        if (!this.isStopped() || !this.isArrived()) {
                this.request();
        }
        if (this.dTravelTime >= this.endTick) {
                RunEnvironment.getInstance().endRun();
        }
        if (!this.isArrived()) {
                if (this.route.atDestination()) {
                        this.setArrived(true);
                        this.route = null;
                        this.setStopped(true);
                        RunEnvironment.getInstance().endRun();
                }
        }
        if (this.route != null && !this.isStopped()) {

                if (!this.bBroken) this.route.travelA(true);
                this.addTravelTime(1.0);
                this.updateRelativeLocation();
        }
   } catch (Exception e) {
        e.printStackTrace();
   }
```

```
}

public void request() {
        requestOfBrokenCar(GlobalVariables.defaultBreakdownResponseRange,
                                GlobalVariables.extendedBreakdownResponseRange);
}

public void requestOfBrokenCar(double defaultBreakdownResponseRange,
                                        double extendedBreakdownResponseRange) {
        this.numRequest = 0;
        this.numRequestOut = 0;
        PrivateVehicle.resetTotalNumStoppedPerTick();
        double a_x, a_y, centre_x, centre_y;
        centre_x = this.getCurrentCoordinateX();
        centre_y = this.getCurrentCoordinateY();
        Envelope env = (Envelope)this.getCurrentGeometry().buffer(
                extendedBreakdownResponseRange * 1.5).getEnvelopeInternal();
        Iterable<PrivateVehicle> vehicleIt = this.getObjects(env, PrivateVehicle.class);
        for (PrivateVehicle v:vehicleIt) {
                if (v.getPreferedLane() != 1) continue;
                if (PrivateVehicle.inTheVehiclePassedByList(v)) {
                        continue;
                }
                a_x = v.getCurrentCoordinateX();
                a_y = v.getCurrentCoordinateY();
                double distance = getDistance(a_x, a_y, centre_x, centre_y);
                if (v.veinOnt.getCurrentRoadEdge() == null) v.updateRelativeLocation();
                if (!v.isBUseComm() && distance <= defaultBreakdownResponseRange) {

                        if (v.veinOnt.isInTheSituationOfVehicleStationary(this.veinOnt,
                                        defaultBreakdownResponseRange)) {
                               v.getRequest(this.getVeinOnt(), this.getCurrentCoordinate());
                                this.numRequest ++;
                                addRequestedVehicleToTheList(v);
                        }
                } else if (v.isBUseComm() &&
                        distance <= extendedBreakdownResponseRange) {
                        if (v.veinOnt.isInTheSituationOfVehicleStationary(
                                this.veinOnt, extendedBreakdownResponseRange)) {
                               v.getRequest(this.getVeinOnt(), this.getCurrentCoordinate());
                                this.numRequest ++;
                                addRequestedVehicleToTheList(v);
                        }
                }
        }
        System.out.println(this.getName() +" => Requested  "
         + this.getNumOfVehicleRequested() + ","
         + this.getNumOfDSRCVehicleRequested() + ","
         + this.getNumOfNonDSRCVehicleRequested() + "||PassBy "
         + this.getNumPassedByPrivateVehicle() + ","
         + this.getNumPassedByWithDSRCPrivateVehicle() + ","
         + this.getNumPassedByWithoutDSRCPrivateVehicle() + "||Stopped "
         + this.getNumOfVehicleStopped() + ","
         + this.getNumOfDSRCVehicleStopped() + ","
         + this.getNumOfNonDSRCVehicleStopped() + "||PassBy2 "
         + this.getNumOfVehiclePassedBy() + ","
         + this.getNumOfDSRCVehiclePassedBy() + ","
```

```
			+ this.getNumOfNonDSRCVehiclePassedBy() + "|| Stopped2 "
			+ PrivateVehicle.totalNumStopped + ","
			+ PrivateVehicle.totalStoppedInterval  + ","
			+ PrivateVehicle.totalCountInfluence + " || Breakdown "
			+ this.getCurrentCoorinatesString()
			);
	}

	public int getNumRequest() {
		return numRequest;
	}

	public int getNumRequestOut() {
		return numRequestOut;
	}

	public int getNumStoppedPrivateVehiclePerTick() {
		return PrivateVehicle.getTotalNumStoppedPerTick();
	}

	public int getNumPassedByPrivateVehicle () {
		return PrivateVehicle.getTotalNumPassedBy();
	}

	public int getNumPassedByWithDSRCPrivateVehicle () {
		return PrivateVehicle.getTotalNumPassedByWithDSRC();
	}

	public int getNumPassedByWithoutDSRCPrivateVehicle () {
		return PrivateVehicle.getTotalNumPassedByWithoutDSRC();
	}

	public int getNumOfVehicleStopped() {
		return  PrivateVehicle.getNumOfVehicleStopped();
	}

	public int getNumOfDSRCVehicleStopped() {
		return PrivateVehicle.getNumOfDSRCVehicleStopped();
	}
	public int getNumOfNonDSRCVehicleStopped() {
		return PrivateVehicle.getNumOfNonDSRCVehicleStopped();
	}

	public int getNumOfVehiclePassedBy() {
		return PrivateVehicle.getNumOfVehiclePassedBy();
	}

	public int getNumOfDSRCVehiclePassedBy() {
		return PrivateVehicle.getNumOfDSRCVehiclePassedBy();
	}
	public int getNumOfNonDSRCVehiclePassedBy() {
		return PrivateVehicle.getNumOfNonDSRCVehiclePassedBy();
	}

	public int getNumOfVehicleRequested() {
		return lstDSRCVehicleRequested.size() + lstNonDSRCVehicleRequested.size();
	}
	public int getNumOfDSRCVehicleRequested() {
		return lstDSRCVehicleRequested.size();
	}
	public int getNumOfNonDSRCVehicleRequested() {
		return lstNonDSRCVehicleRequested.size();
	}

	public void addRequestedVehicleToTheList(PrivateVehicle pv) {
		String vehicleName = pv.getName();
		if (pv.isBUseComm()) {
			if (!lstDSRCVehicleRequested.contains(vehicleName))
				lstDSRCVehicleRequested.add(vehicleName);
		} else {
			if (!lstNonDSRCVehicleRequested.contains(vehicleName))
				lstNonDSRCVehicleRequested.add(vehicleName);
		}
	}

	public boolean isBroken() {
		return bBroken;
	}

	public void setBroken(boolean broken) {
		this.bBroken = broken;
	}

	public int getEndTick() {
		return endTick;
	}

	public void setEndTick(int endTick) {
		this.endTick = endTick;
	}
}
```

D.3.4 PrivateVehicle.java

```
public class PrivateVehicle extends Vehicle {

	private static int totalNumStoppedPerTick = 0;
	protected static int totalNumStopped = 0;
	protected static double totalCountInfluence = 0;
	protected static double totalStoppedInterval = 0.0;
	private static ArrayList<String> lstDSRCVehiclePassedBy;
	private static ArrayList<String> lstNonDSRCVehiclePassedBy;
	private static ArrayList<String> lstDSRCVehicleStopped;
	private static ArrayList<String> lstNonDSRCVehicleStopped;
	private static int totalNumPassedBy = 0;
	private static int totalNumPassedByWithDSRC = 0;
	private static int totalNumPassedByWithoutDSRC = 0;
	private int numRerouted;
	private boolean bUseComm;
private boolean bGetRequested;
private boolean bIgnoreTheSituationSignOn;
double refDistance1, refDistance2, refDistance3;
CommunicationMessage emergencyVehicleVeinOnt;
Coordinate emergencyVehicleCoord;

public PrivateVehicle(Infra veinInfra, String vehicleType, boolean bUseComm) {
	super(veinInfra, vehicleType, bUseComm);
	this.numRerouted = 0;
	this.bUseComm = bUseComm;
	this.bIgnoreTheSituationSignOn = false;
	this.emergencyVehicleVeinOnt = null;
	this.refDistance1 = 0;
	this.refDistance2 = 0;
	this.refDistance3 = 0;
	this.emergencyVehicleCoord = null;
}

public static void resetStaticVariables() {
	lstDSRCVehiclePassedBy = null;
	lstNonDSRCVehiclePassedBy = null;
	lstDSRCVehicleStopped = null;
	lstNonDSRCVehicleStopped = null;
	lstDSRCVehiclePassedBy = new ArrayList<String>();
	lstNonDSRCVehiclePassedBy = new ArrayList<String>();
	lstDSRCVehicleStopped = new ArrayList<String>();
	lstNonDSRCVehicleStopped = new ArrayList<String>();
	totalNumStoppedPerTick = 0;
	totalNumStopped = 0;
	totalCountInfluence = 0.0;
	totalStoppedInterval = 0.0;
	totalNumPassedBy = 0;
	totalNumPassedByWithDSRC = 0;
	totalNumPassedByWithoutDSRC = 0;
}

public int getNumRerouted() {
	return numRerouted;
}

public static int getTotalNumStoppedPerTick() {
	return totalNumStoppedPerTick;
}

public static void resetTotalNumStoppedPerTick() {
	totalNumStoppedPerTick = 0;
}

public static int getTotalNumPassedBy() {
	return totalNumPassedBy;
}

public static int getTotalNumPassedByWithDSRC() {
	return totalNumPassedByWithDSRC;
}

public static int getTotalNumPassedByWithoutDSRC() {
```

```
            return totalNumPassedByWithoutDSRC;
}

public static void addTotalNumPassedBy(boolean bUseOnt) {
            if (bUseOnt) totalNumPassedByWithDSRC ++;
            else totalNumPassedByWithoutDSRC ++;
            totalNumPassedBy ++;
}

public boolean isBIgnoreTheSituationSignOn() {
            return bIgnoreTheSituationSignOn;
}

public void setBIgnoreTheSituation(boolean ignoreTheSituation) {
            bIgnoreTheSituationSignOn = ignoreTheSituation;
}

public boolean isBUseComm() {
            return bUseComm;
}

public String getOutputter() {
            return this.getName()+ ","
                        + this.getSpeed() + ","
                        + this.infra.formatStr( this.getCurrentCoordinateX()) + ","
                        + this.infra.formatStr( this.getCurrentCoordinateY()) + ","
                        + this.getCurrentRoadFID() + ","
                        + this.getNextJunctionID() + ","
                        + this.infra.formatStr(this.getRemainDistanceToNextJunction()) + ","
                        + this.bUseComm + ","
                        + this.getDCountInfluence() + ","
                        + this.dStopInterval + ","
                        + bGetRequested + ","
                        + this.isStopped() + ","
                        + this.getDTravelTime(); // + ","     + "x";
}

@Override
public void step() {
   try {
            if (this.route != null) {
                        if (this.route.atDestination()) {
                                    this.route = null;
                                    this.resetPrivateVehicle();
                        }
            }
            if (this.veinOnt.getCurrentRoadEdge() == null)
                        this.updateRelativeLocation();
            if (this.bRequested) {
                        bGetRequested = true;
                        if (this.answerback(this.emergencyVehicleVeinOnt,
                                                                        this.emergencyVehicleCoord)) {
                                    this.addStopInterval(1.0);
                                    totalNumStoppedPerTick++;
                                    totalNumStopped++;
                                    addStoppedVehicleToTheList(this);
                        }
                        this.emergencyVehicleVeinOnt = null;
                        this.emergencyVehicleCoord = null;
                        this.bRequested = false;
                } else if (this.isStopped()) {
                        bGetRequested = false;
                        this.setStopped(false);
                        this.resetStopVariables();
                } else bGetRequested = false;
                        if (this.isStopped()) {}
                        else {
                                this.changeVelocityIDM(0.1);
                                this.route.travelA(true);
                                this.addTravelTime(1.0);
                                this.updateRelativeLocation();
                        }
  } catch (Exception e) {
          e.printStackTrace();
  }
}

public void initialMove() {
  try {
          if (this.route != null) {
                  if (this.route.atDestination()) {
                          this.route = null;
                          this.resetPrivateVehicle();
                  }
          }
          if (this.veinOnt.getCurrentRoadEdge() == null)
                  this.route.travelA(true); // This will move the vehicle towards their destination
  } catch (Exception e) {
                  e.printStackTrace();
  }
}

public void resetStopVariables () {
          this.bIgnoreTheSituationSignOn = false;
          refDistance1 = 0.0;
          refDistance2 = 0.0;
          refDistance3 = 0.0;
}

public void getRequest(CommunicationMessage emergencyVehicleVeinOnt,
                                                Coordinate emergencyVehicleCoord ) {
          this.emergencyVehicleVeinOnt = emergencyVehicleVeinOnt;
          this.emergencyVehicleCoord = emergencyVehicleCoord;
          this.bRequested = true;
}

public boolean answerback(CommunicationMessage emergencyVehicleVeinOnt,
                                                Coordinate emergencyVehicleCoord) {
          if (GlobalVariables.iCase < 4)
            return answerback2Ambulance(emergencyVehicleVeinOnt, emergencyVehicleCoord);
          else if (GlobalVariables.iCase == 4)
            return answerback2Breakdown(emergencyVehicleVeinOnt, emergencyVehicleCoord);
          return true;
```

```
}

public boolean answerback2Breakdown(CommunicationMessage breakdownOnt,
                                                        Coordinate breakdownVehicleCoord) {
        boolean flag = false;
        if (this.getPreferedLane() != 1) return flag;
        double remainingDistanceToTheBreakdown
                        = this.veinOnt.getRemainingDistanceToBreakdown(breakdownOnt);
        double remainingTimeToTheBreakdown
                        = this.veinOnt.getRemainingTimeToBreakdown(breakdownOnt);
        double remainingTimeToTheBreakdownV0 = remainingDistanceToTheBreakdown
                         / (this.v0 * GlobalVariables.ConversionCoefficient4MeterPerSec);
        if (this.speed > 10 && remainingDistanceToTheBreakdown > this.v0maxLane1
                                        * this.conversionMph2Ms * 2.0) { //60mph * 2sec
                if(this.fnIsSafeToOverTakeTheBreakdown(4.0)) {
                        this.moveOverToTheRightLane();
                        addTotalNumPassedBy(this.bUseComm);
                        addPassedByVehicleToTheList(this);
                }
        } else {
                this.bSuddenDeceleration = true;
                PrivateVehicle vlist[];
                Double s = -100.0;
                try {
                        vlist = this.fnSearchFrontAndBehindVehicles(
                        GlobalVariables.searchRange4IDM, this.getPreferedLane(), false);
                } catch (Exception e) {
                        vlist = null;
                }

                if(vlist[0]!=null)  {
                        s = this.veinOnt.getRemainingDistanceToTheVehicleAhead(
                                                                vlist[0].veinOnt); //m
                }
                if (s > 0 && s < remainingDistanceToTheBreakdown) {
                        this.decelerationIDM (this.speed, s, 0.1);
                        if (s < Vehicle.vlength/2.0) this.decelerate();
                } else {
                        this.decelerationIDM (this.speed,
                                                remainingDistanceToTheBreakdown, 0.1);
                        if (remainingDistanceToTheBreakdown < 3*Vehicle.vlength)
                                                                this.decelerate();
                }
                        if(s < 0 && this.fnIsSafeToOverTakeTheBreakdown(2.0 + 4.0)) {
                                        this.moveOverToTheRightLane();
                        } else {
                         if ((s < 0 || (s > Vehicle.vlength/2.0)) && this.speed < 10
                         && remainingDistanceToTheBreakdown > 3*Vehicle.vlength)
                                                                this.speed = 10;
                         this.addInfluenceTime(1.0);
                         flag = true;
                        }
        }
        return flag;
}

public boolean answerback2Ambulance(CommunicationMessage emergencyVehicleVeinOnt,
                                                Coordinate emergencyVehicleCoord) {
        boolean flag = false;
        if (this.bUseComm) {
                if (emergencyVehicleVeinOnt == null) return false;
                if (this.getVeinOnt().isInTheSituationOfEmergencyVehicle(
                                                        emergencyVehicleVeinOnt)) {
                        this.bIgnoreTheSituationSignOn = false;
                        if (this.isStopped()) {
                        } else {
                                this.setStopped(true);
                                this.addInfluenceTime(1.0);
                        }
                        flag = true;
                } else {
                        this.bIgnoreTheSituationSignOn = true;
                        if (this.isStopped()) {
                                this.setStopped(false);
                                addTotalNumPassedBy(this.bUseComm);
                                addPassedByVehicleToTheList(this);
                        }
                }
        } else {
                if (this.emergencyVehicleCoord == null) return false;
                Coordinate thisCoord
                        = infra.getRoadGeography().getGeometry(this).getCoordinate();
                if (this.getVeinOnt().IsInTheAmbulanceSituationWithSirenRange2(
                        emergencyVehicleVeinOnt, emergencyVehicleCoord, thisCoord)) {
                        if (this.isStopped()) {}
                        else {
                                this.setStopped(true);
                                this.addInfluenceTime(1.0);
                        }
                        flag = true;
                } else {
                        if (this.isStopped()) {
                                this.setStopped(false);
                                addTotalNumPassedBy(this.bUseComm);
                                addPassedByVehicleToTheList(this);
                        }
                }
        }
        return flag;
}

public int changeVelocityIDM(double unitsec) {
        try {
                if(this.bSuddenDeceleration) return 0;
                if (this.route.getRouteCoords().size() < 2) return 1;
                if (this.route.atDestination()) return 2;
                double v0 = this.v0 * conversionMph2Ms; //m/s
                double v = this.getSpeed() * conversionMph2Ms; //m/s
                double dv;
                double s = 0; //m
                double sFront, sBehind;
                double sstar;
                double s1;
                double va;
                double vafree;
                double vaint;
                PrivateVehicle vlist[];
                try {
                        vlist = this.fnSearchFrontAndBehindVehicles(
                          GlobalVariables.searchRange4IDM, this.getPreferedLane(), false);
                } catch (Exception e) {
                        vlist = null;
                        return 3;
                }
                if(vlist[1]!=null) {
                        sBehind = vlist[1].veinOnt
                            .getRemainingDistanceToTheVehicleAhead(this.veinOnt); //m
                        if(sBehind > 0 && sBehind < (this.T + 0.5) * v0) {
                         if (this.getPreferedLane() == 1
                                 && v0 < this.v0maxLane1 * conversionMph2Ms)
                            v0 = this.v0maxLane1 * conversionMph2Ms;//(v0 +
                                          this.v0maxLane1 * conversionMph2Ms) / 2.0;
                         else if (this.getPreferedLane() == 2
                                 && v0 < this.v0maxLane2 * conversionMph2Ms)
                            v0 = this.v0maxLane2 * conversionMph2Ms;
                         else if (this.getPreferedLane() == 3
                                 && v0 < this.v0maxLane3 * conversionMph2Ms)
                            v0 = this.v0maxLane3 * conversionMph2Ms;
                        }
                }
                vafree = this.a * (1.0 - Math.pow(v/v0, this.delta)) ;
                if(vlist[0]==null) va = vafree;
                else {
                        s = this.veinOnt
                          .getRemainingDistanceToTheVehicleAhead(vlist[0].veinOnt); //m
                        dv = v - vlist[0].veinOnt.getSpeed()*conversionMph2Ms;
                        s1 = v*this.T + v*dv / (2.0*Math.sqrt(this.a + this.b));
                        if (s1 > 0) sstar = this.S0 + s1;
                        else sstar = this.S0;
                        vaint = -1.0 * this.a * Math.pow(sstar/s,2);
                        va = vafree + vaint * unitsec ;
                }
                v = (v + va) * conversionMs2Mph; // va 1m/s=3.6kmh, kmh to mph
                if (v < 0) v = 0;
                this.setSpeed(v);
                if(vlist[0]!=null &&  s < this.speed *
                        GlobalVariables.ConversionCoefficient4MeterPerSec * this.T{
                        this.decelerate(1);
                }
                vlist = null;
                return (int) v;

                } catch (Exception e) {
                        e.printStackTrace();
                }
                return 5;
 }

public PrivateVehicle[] fnSearchFrontAndBehindVehicles(double searchDistanceRange, int lane,
                                                                        boolean bStr) {
        PrivateVehicle vlist[] = new PrivateVehicle[2];
```

```
		vlist[0] = null;
		vlist[1] = null;
		try {
		double relativeDistance;
		double absoluteFrontSearchDistance;
		double absoluteBehindSearchDistance;
		double frontDist;
		double behindDist;
		double a_x, a_y, centre_x, centre_y;
		centre_x = this.getCurrentCoordinateX();
		centre_y = this.getCurrentCoordinateY();
		Envelope env = (Envelope)this.getCurrentGeometry().buffer(
							searchDistanceRange * 1.5).getEnvelopeInternal();
		Iterable<PrivateVehicle> vehicleIt = this.getObjects(env, PrivateVehicle.class);
		absoluteFrontSearchDistance = searchDistanceRange;
		absoluteBehindSearchDistance = searchDistanceRange;
		double distance;
		int i=0;
		for (PrivateVehicle v:vehicleIt) {
			i++;
			if (this.getName().compareTo(v.getName()) == 0) continue;
			if (v.getPreferedLane() != lane) continue;
			a_x = v.getCurrentCoordinateX();
			a_y = v.getCurrentCoordinateY();
			distance = getDistance(a_x, a_y, centre_x, centre_y);
			if (v.veinOnt.getCurrentRoadEdge() == null) v.updateRelativeLocation();
			relativeDistance
			    = v.veinOnt.getRemainingDistanceToTheVehicleAhead(this.veinOnt);
			if (relativeDistance > 0.0) {
				if (relativeDistance < absoluteBehindSearchDistance) {
					vlist[1] = v;
					absoluteBehindSearchDistance = relativeDistance;
					continue;
				}
			} else {
				relativeDistance = this.veinOnt
					.getRemainingDistanceToTheVehicleAhead(v.veinOnt);
				if (relativeDistance > 0.0) {
					if (relativeDistance < absoluteFrontSearchDistance) {
					    vlist[0] = v;
					    absoluteFrontSearchDistance = relativeDistance;
					    continue;
					}
				}
			}
		}
		if (vlist[0]!=null && this.veinOnt.getRemainingDistanceToTheVehicleAhead(
		    vlist[0].veinOnt) > 0 && this.veinOnt.getRemainingDistanceToTheVehicleAhead(
		    vlist[0].veinOnt) <= Vehicle.vlength) {
			this.decelerate(1);
		}
		return vlist;
		} catch (Exception e) {
			e.printStackTrace();
		}
		return vlist;
	}
```

```
public boolean fnIsSafeToOverTakeTheBreakdown(double safeTimeRange) {  //four seconds
        boolean flag;
        double aRelativeDistance = -1;
        double absoluteSafeDistance = -1;
        double aRelativeDistance4frontVehicle = -1;
        double absoluteSafeDistance4frontVehicle = -1;
        double myRelativeDistance;
        double myRelativeDistance4frontVehicle;

        flag = true;
        myRelativeDistance = 3000;
        myRelativeDistance4frontVehicle = 3000;
        double a_x, a_y, centre_x, centre_y;
        centre_x = this.getCurrentCoordinateX();
        centre_y = this.getCurrentCoordinateY();
        Envelope env = (Envelope)this.getCurrentGeometry().buffer(
           GlobalVariables.defaultBreakdownResponseRange * 1.5).getEnvelopeInternal();
        Iterable<PrivateVehicle> vehicleIt = this.getObjects(env, PrivateVehicle.class);

        for (PrivateVehicle v:vehicleIt) {
                if (v.getPreferedLane() != 2) continue;
                a_x = v.getCurrentCoordinateX();
                a_y = v.getCurrentCoordinateY();
                double distance = getDistance(a_x, a_y, centre_x, centre_y);
                if (v.veinOnt.getCurrentRoadEdge() == null) v.updateRelativeLocation();
                absoluteSafeDistance4frontVehicle = Vehicle.T  * v.speed *
                        GlobalVariables.ConversionCoefficient4MeterPerSec + this.vlength;
                aRelativeDistance4frontVehicle
                        = this.veinOnt.getRemainingDistanceToFrontVehicle(v.veinOnt);
                if (aRelativeDistance4frontVehicle > 0.0 &&
                   aRelativeDistance4frontVehicle < myRelativeDistance4frontVehicle &&
                   aRelativeDistance4frontVehicle <  absoluteSafeDistance4frontVehicle) {
                        myRelativeDistance4frontVehicle = aRelativeDistance4frontVehicle;
                        flag = false;
                        return flag;
                } else if (Math.abs(aRelativeDistance4frontVehicle) < this.vlength )  {
                        flag = false;
                        return flag;
                }
                absoluteSafeDistance = safeTimeRange * v.speed
                        * GlobalVariables.ConversionCoefficient4MeterPerSec - this.vlength;
                aRelativeDistance
                   = this.veinOnt.getRemainingDistanceToTheVehicleBehind(v.veinOnt);
                if (aRelativeDistance > 0.0 && aRelativeDistance < myRelativeDistance
                                        && aRelativeDistance < absoluteSafeDistance ) {
                        myRelativeDistance = aRelativeDistance;
                        flag = false;
                        return flag;
                } else if (Math.abs(aRelativeDistance) < this.vlength )  {
                        flag = false;
                        return flag;
                }
        }
        return flag;
}
```

```
public static int getNumOfVehiclePassedBy() {
		return lstDSRCVehiclePassedBy.size() + lstNonDSRCVehiclePassedBy.size();
}

public static int getNumOfDSRCVehiclePassedBy() {
		return lstDSRCVehiclePassedBy.size();
}
public static int getNumOfNonDSRCVehiclePassedBy() {
		return lstNonDSRCVehiclePassedBy.size();
}

public static void addPassedByVehicleToTheList(PrivateVehicle pv) {
		String vehicleName = pv.getName();
		if (pv.isBUseComm()) {
			if (!lstDSRCVehiclePassedBy.contains(vehicleName))
				lstDSRCVehiclePassedBy.add(vehicleName);
		} else {
			if (!lstNonDSRCVehiclePassedBy.contains(vehicleName))
				lstNonDSRCVehiclePassedBy.add(vehicleName);
		}
}

public static boolean inTheVehiclePassedByList (PrivateVehicle pv) {
		String vehicleName = pv.getName();
		boolean myReturn = false;
		if (pv.isBUseComm()) {
			if (lstDSRCVehiclePassedBy.contains(vehicleName)) myReturn = true;
		} else {
			if (lstNonDSRCVehiclePassedBy.contains(vehicleName)) myReturn = true;
		}
		return myReturn;
}

public static int getNumOfVehicleStopped() {
		return lstDSRCVehicleStopped.size() + lstNonDSRCVehicleStopped.size();
}

public static int getNumOfDSRCVehicleStopped() {
		return lstDSRCVehicleStopped.size();
}
public static int getNumOfNonDSRCVehicleStopped() {
		return lstNonDSRCVehicleStopped.size();
}

public static void addStoppedVehicleToTheList(PrivateVehicle pv) {
		String vehicleName = pv.getName();
		if (pv.isBUseComm()) {
			if (!lstDSRCVehicleStopped.contains(vehicleName))
				lstDSRCVehicleStopped.add(vehicleName);
		} else {
			if (!lstNonDSRCVehicleStopped.contains(vehicleName))
				lstNonDSRCVehicleStopped.add(vehicleName);
		}
}

public void createNewPrivateVehicle () {
			boolean  bUseOnt = false;
```

```
                    int iUseOnt = this.infra.random(0, 100, false); //true
                    if (iUseOnt < GlobalVariables.iPercentUsingComm) bUseOnt = true;
                    else if (iUseOnt == 100 && GlobalVariables.iPercentUsingComm == 100)
                                                                             bUseOnt = true
                    PrivateVehicle pVehicle = new PrivateVehicle(
                                                  this.infra, "PrivateVehicle", bUseOnt);
                    pVehicle.setId(this.infra.getNextVehicleNumber());
                    pVehicle.setName("V_" + pVehicle.getId());
                    double dspeed = GlobalVariables.TRAVEL_SPEED_MPH ; // 3.5;
                    int add = this.infra.random (-30, 30, false);
                    double speed = dspeed  + dspeed * add / 100.0;
                    pVehicle.setSpeed(speed);
                    this.infra.getRoadContext().add(pVehicle);
                    pVehicle.setOriginJuncRandomly();
                    pVehicle.initializeRoute(pVehicle.getOriginJunc(), null, null);
                    pVehicle.updateRelativeLocation();
                    ISchedule schedule = RunEnvironment.getInstance().getCurrentSchedule();
                    ScheduleParameters endParams
                      = ScheduleParameters.createAtEnd(ScheduleParameters.LAST_PRIORITY);
                    schedule.schedule(endParams, this, "end");
                    ScheduleParameters startParams = ScheduleParameters.createOneTime(1);
                    schedule.schedule(startParams, this, "start");
                    ScheduleParameters agentParams = ScheduleParameters.createRepeating(
                                                                                  1, 1, 0);
                    schedule.schedule(agentParams, pVehicle, "step");
}

public void resetPrivateVehicle () {
          boolean  bUseOnt = false;
          int iUseOnt = this.infra.random(0, 100, false); //true
          if (iUseOnt < GlobalVariables.iPercentUsingComm) bUseOnt = true;
          else if (iUseOnt == 100 && GlobalVariables.iPercentUsingComm == 100)
                                                                             bUseOnt = true;
          this.bStopped = false;
          this.bArrived = false;
          this.dTravelTime = 0.0;
          this.dCountInfluence = 0.0;
          this.dStopInterval = 0.0;
          this.bRequested = false;
          this.numRerouted = 0;
          this.bUseComm = bUseOnt;
          this.bIgnoreTheSituationSignOn = false;
          this.emergencyVehicleVeinOnt = null;
          this.refDistance1 = 0;
          this.refDistance2 = 0;
          this.refDistance3 = 0;
          this.emergencyVehicleCoord = null;
          this.setId(this.infra.getNextVehicleNumber());
          this.setName("V_" + this.getId());
          int iSpeedRandom = this.infra.random(0, 100, false);
          this.setSpeedAndPreferedLane(iSpeedRandom);
          if (GlobalVariables.iCase == 4) {
                    this.setOriginJuncRandomlyFromRandomHelper(4);
                    this.initializeRouteFromRandomHelper(null, null, null, 4);
          }
          else {
                    this.initializeRoute(null, null, null);
```

```
                }
                this.updateRelativeLocation();
        }

        public String toString() {
                return this.getName() + ", " + this.getDTravelTime() + ", " + this.bUseComm + ": " ;
        }
}
```

D.3.5 Route.java

```
public class Route {

        private Vehicle vehicle;
        private Network roadNetwork;
        private Geography roadGeography;
        private Infra infra;
        private Junction originJunc;
        private Junction destJunc;
        private List<Coordinate> routeCoords;
        private ArrayList<RoadEdge> routeEdges;
        private ArrayList<Junction> routeJunctions;
        private GeometryFactory geomFac;
        private double travelPerTurn;
        private double little_buffer_distance;
        private double big_buffer_distance;

        public Route(Infra mainInfra, Vehicle vehicle, Junction originJunction, Junction destJunction)
                                                                                throws Exception {
                this.vehicle = vehicle;
                this.originJunc = originJunction;
                this.destJunc = destJunction;
                if (this.vehicle==null || this.originJunc==null  || this.destJunc==null) {
                  throw new NullPointerException("Route() error: one of the input parameters is null");
                }
                this.infra = mainInfra;
                this.roadNetwork = infra.getRoadNetwork();
                this.roadGeography = infra.getRoadGeography();
                this.routeCoords = new ArrayList<Coordinate>();
                this.routeEdges = new ArrayList<RoadEdge>();
                this.routeJunctions = new ArrayList<Junction>();
                this.geomFac = new GeometryFactory();
                this.travelPerTurn = GlobalVariables.TRAVEL_PER_TURN;
                this.little_buffer_distance = 0.0001;
                this.big_buffer_distance = 100;
                Coordinate c = this.originJunc.getCoordinate();
                Point p = new GeometryFactory().createPoint(c);
                this.roadGeography.move(vehicle, p);
                double time;
                time = System.currentTimeMillis();
                routeCoords.addAll(getRouteBetweenJunctions(originJunc, destJunc));
                removeDuplicateCoords();
        }
```

```
public void kill() {
	this.vehicle = null;
	this.originJunc = null;
	this.destJunc = null;
	this.infra = null;
	this.roadNetwork = null;
	this.roadGeography = null;
	this.routeCoords.clear();
	this.routeCoords = null;
	this.routeEdges.clear();
	this.routeEdges = null;
	this.routeJunctions.clear();
	this.routeJunctions = null;
	this.geomFac = null;
}

public boolean travelA(boolean smoothMoving) {
	if (this.vehicle.getSpeed()==0) return false;
	this.travelPerTurn = this.vehicle.getSpeed() *
	    GlobalVariables.TRAVEL_PER_TURN/GlobalVariables.TRAVEL_SPEED_MPH;
	this.travelPerTurn = this.travelPerTurn / 10.0;  //1 tick = 0.1 sec
	if (atDestination()) {
		return false;
	}
	double time; // used for debugging
	double distTravelled = 0; // The distance travelled so far
	Coordinate currentCoord = null;      // Current location
	Coordinate target = null;            // Target coordinate we're heading for (in route list)
	boolean travelledMaxDist = false;   // True when travelled maximum dist this iteration
	DecimalFormat df = new DecimalFormat("#.########");
	int i = 0;
	while (!travelledMaxDist && !atDestination() ) {
		i++;
		currentCoord = roadGeography.getGeometry(this.vehicle).getCoordinate();
		target = routeCoords.get(0);
		Geometry currentGeom = geomFac.createPoint(currentCoord);
		Geometry targetGeom = geomFac.createPoint(target);
		double distToTarget = DistanceOp.distance(currentGeom, targetGeom);
		double angle;
		double distToTravel;
		double regdist, dx, dy;
		angle = angle(target, currentCoord)+Math.PI;
		regdist = this.travelPerTurn / 10.0;
		if (distTravelled+distToTarget < travelPerTurn) {
			distTravelled += distToTarget;
			if (smoothMoving) roadGeography.move(vehicle, targetGeom);
			routeCoords.remove(0);
		} // if
		else {
			angle = angle(target, currentCoord)+Math.PI;
			distToTravel = travelPerTurn-distTravelled;
			dx = distToTravel *  Math.cos(angle);
			dy = distToTravel *  Math.sin(angle);
			roadGeography.moveByDisplacement(vehicle, dx, dy);
			travelledMaxDist = true;
		} // else
```

```
			} // while
			return true;
	}

	public boolean travelA(int lane, boolean smoothMoving) {
			double dShift = 0.0;
			dShift= (double) (lane - 3);
			this.travelPerTurn = this.vehicle.getSpeed()
			  *  GlobalVariables.TRAVEL_PER_TURN/GlobalVariables.TRAVEL_SPEED_MPH;
			this.travelPerTurn = this.travelPerTurn / 10.0;  ///08 1 tick = 0.1 sec
			if (atDestination()) {
					return false;
			}
			double time; // used for debugging
			double distTravelled = 0; // The distance travelled so far
			Coordinate currentCoord = null;      // Current location
			Coordinate target = null;                 // Target coordinate we're heading for (in route list)
			boolean travelledMaxDist = false;   // True when travelled maximum dist this iteration
			DecimalFormat df = new DecimalFormat("#.########");
			int i = 0;
			while (!travelledMaxDist && !atDestination() ) {
					i++;
					currentCoord = roadGeography.getGeometry(this.vehicle).getCoordinate();
					target = routeCoords.get(0);
					Geometry currentGeom = geomFac.createPoint(currentCoord);
					Geometry targetGeom = geomFac.createPoint(target);
					double distToTarget = DistanceOp.distance(currentGeom, targetGeom);
					double angle;
					double distToTravel;
					double regdist, dx, dy;
					angle = angle(target, currentCoord)+Math.PI;
					regdist = this.travelPerTurn / 10.0;
					if (distTravelled+distToTarget < travelPerTurn) {
							distTravelled += distToTarget;
							if (smoothMoving) roadGeography.move(vehicle, targetGeom);
							routeCoords.remove(0);
					} // if
					else {
							angle = angle(target, currentCoord)+Math.PI;
							distToTravel = travelPerTurn-distTravelled;
							dx = distToTravel *  Math.cos(angle);
							dy = distToTravel *  Math.sin(angle);
							roadGeography.moveByDisplacement(vehicle, dx, dy);
							travelledMaxDist = true;
					} // else
			} // while
			return true;
	}

	private ArrayList<Coordinate> getCoordsAlongRoad(Coordinate currentCoord,
	 Coordinate destinationCoord, RoadElement road, boolean toJunction) throws Exception {
			double time = System.currentTimeMillis();
			Coordinate[] roadCoords = this.roadGeography.getGeometry(road).getCoordinates();
			ArrayList<Coordinate> routeCoords = new ArrayList<Coordinate>();
			boolean currentCorrect = false, destinationCorrect= false;;
			for (int i=0; i<roadCoords.length; i++) {
					if (toJunction && roadCoords[i].equals(destinationCoord)) {
						destinationCorrect = true;
						break;
					}
					else if (!toJunction  && roadCoords[i].equals(currentCoord)) {
						currentCorrect = true;
						break;
					}
				} // for
				if (!(destinationCorrect || currentCorrect)) {
					throw new Exception("Route: getCoordsAlongRoad: Error" );
				}
				// Might need to reverse the order of the road coordinates
				if (toJunction && !roadCoords[roadCoords.length-1].equals(destinationCoord)) {
					ArrayUtils.reverse(roadCoords);
				}
				else if (!toJunction && !roadCoords[0].equals(currentCoord)){
					ArrayUtils.reverse(roadCoords);
				}
				Point destinationPointGeom = geomFac.createPoint(destinationCoord);
				Point currentPointGeom = geomFac.createPoint(currentCoord);
				for (int i=0; i<roadCoords.length-1; i++ ) {
					Coordinate[] segmentCoords = new Coordinate[]{
										roadCoords[i], roadCoords[i+1]};
					LineString segment = geomFac.createLineString(segmentCoords);
					Geometry buffer = segment.buffer(little_buffer_distance);
					if (!toJunction) {
						routeCoords.add(roadCoords[i]);
						if (destinationPointGeom.within(buffer)) {
							routeCoords.add(destinationCoord);
							return routeCoords;
						}
					}
					else if (toJunction) {
						// coords which make up the road segment
						if (currentPointGeom.within(buffer)) {
							routeCoords.add(destinationCoord);
							for (int j=i+1; j<roadCoords.length; j++) {
								routeCoords.add(roadCoords[j]);
							}
							return routeCoords;
						}
					}
				} // for
				// If we get here then the route hasn't been created
				return null;
			}

			private ArrayList<Coordinate> getRouteBetweenJunctions (
									Junction fromJunc, Junction toJunc) {
				double time = System.currentTimeMillis();
				List<RepastEdge<Junction>> shortestPath
					= new MyShortestPath<Junction>(roadNetwork).getPath(fromJunc, toJunc);
				if (shortestPath == null) return null;
				boolean bFlip = false;
				routeJunctions.add(fromJunc);
				for (RepastEdge<Junction> edge:shortestPath) {
					MyRepastEdge myEdge = (MyRepastEdge) edge;
```

```
}

            String edgeID = myEdge.getEdgeID();
            if (edgeID.endsWith("-")) bFlip = true;
            else bFlip = false;
            String roadID = edgeID.substring(0, edgeID.length()-1);
            RoadEdge roadEdge = infra.getRoadEdgeWithID(roadID, bFlip);
            Junction roadJunction;
            if (bFlip) roadJunction = roadEdge.getRoadElement().getFromJunc();
            else roadJunction = roadEdge.getRoadElement().getToJunc();
            routeEdges.add(roadEdge);
            routeJunctions.add(roadJunction);
        }

        ArrayList<Coordinate> coordPath = new ArrayList<Coordinate>();
        for (RoadEdge roadEdge:routeEdges) {
            RoadElement road = roadEdge.getRoadElement();
            Coordinate[] coords = roadGeography.getGeometry(road).getCoordinates();
            if (roadEdge.isFlipped()) {
                ArrayUtils.reverse(coords);
            }
            for (Coordinate coord:coords) {
                coordPath.add(coord);
            } // for coord:coords
        } // for road:roadPath
        return coordPath;
    }

    public Coordinate getRandomRoadCoordinate() {
        RoadElement road
            = (RoadElement) infra.getRoadContext().getRandomObjects(RoadElement.class, 1);
        Geometry roadGeom = roadGeography.getGeometry(road);
        Coordinate[] coords = roadGeom.getCoordinates();
        return coords[RandomHelper.nextIntFromTo(0,coords.length-1)];
    }

    public boolean atDestination() {
        double time = System.currentTimeMillis();
        if (this.roadGeography.getGeometry(this.vehicle).getCoordinate()
                                        .equals(this.destJunc.getCoordinate())) {
            return true;
        }
        return false;
    }

    private boolean onRoad(Coordinate coord) {
        return infra.onRoad(coord);
    }

    private void removeDuplicateCoords() {
        LinkedHashSet<Coordinate> set = new LinkedHashSet<Coordinate>();
        for (Coordinate coord:routeCoords)
            set.add(coord);
        this.routeCoords = new Vector<Coordinate>();
        for (Coordinate coord:set) {
            this.routeCoords.add(coord);
        }
    }
```

```
public static double convertToMeters(double dist) {
            double distInMeters
                        = NonSI.NAUTICAL_MILE.getConverterTo(SI.METER).convert(dist*60);
            return distInMeters;
}

public static double convertFromMeters(double dist) {
            double distInDegrees
                        =  SI.METER.getConverterTo(NonSI.NAUTICAL_MILE).convert(dist)/(60.0);
            return distInDegrees;
}

public static double distanceM (Coordinate c1, Coordinate c2) {
            return 0;
}

public static double angle(Coordinate p0, Coordinate p1) {
      double dx = p1.x - p0.x;
      double dy = p1.y - p0.y;
      return Math.atan2(dy, dx);
}

public Coordinate getDestination() {
            return this.destJunc.getCoordinate();
}

public List<Coordinate> getRouteCoords() {
            return routeCoords;
}

public double getTotalLenghtOfTheRoute() {
            int loop = routeCoords.size();
            double dist;
            double totalDist = 0.0;
            Coordinate c1 = routeCoords.get(0);
            Coordinate c2;
            Point p1, p2;
            p1 = new GeometryFactory().createPoint(c1);
            for (int i = 1; i < loop; i++) {
                        c2 = routeCoords.get(i);
                        p2 = new GeometryFactory().createPoint(c2);
                        dist = p1.distance(p2);
                        totalDist += dist;
                        p1 = null;
                        p2 = null;
                        c1 = routeCoords.get(i);
                        p1 = new GeometryFactory().createPoint(c1);
            }
            p1 = null;
            p2 = null;
            return totalDist;
}

public boolean checkRouteCoords() {
            boolean bCheck = false;
            Coordinate destCoord1 = this.destJunc.getCoordinate();
            Coordinate destCoord2;  // = null;
```

```
		if (routeCoords.size() > 0) destCoord2 = routeCoords.get(routeCoords.size() - 1);
		else return false;
		if (destCoord1.equals(destCoord2))		bCheck = true;
		else {
			bCheck = false;
		}
		return bCheck;
	}

	public ArrayList<RoadEdge> getRouteEdges() {
		return routeEdges;
	}

	public ArrayList<Junction> getRouteJunctions() {
		return routeJunctions;
	}
}
```

D.3.6 MyShortestPath.java

```
public class MyShortestPath<T> implements ProjectionListener<T> {

	private Network<T> net;
	private boolean calc = true;
	private T source;
	private JungEdgeTransformer transformer;
	private DijkstraShortestPath<T,RepastEdge<T>> dsp;

	public MyShortestPath(Network<T> net){
		init(net);
	}

	@Deprecated
	public MyShortestPath(Network<T> net, T source) {
		this.source = source;
	  init(net);
	}

	private void init(Network<T> net){
		this.net = net;
		transformer = new JungEdgeTransformer<T>();
		net.addProjectionListener(this);
	}

	public List<RepastEdge<T>> getPath(T source, T target){
		if (calc){
			calcPaths();
			calc = false;
		}
		return dsp.getPath(source, target);
	}

	public double getPathLength(T source, T target){
		if (calc){
			calcPaths();
			calc = false;
		}
		Number n = dsp.getDistance(source, target);
		if (n != null)
			return n.doubleValue();
		else
			return Double.POSITIVE_INFINITY;
	}

	@Deprecated
	public double getPathLength(T target){
		return getPathLength(this.source, target);
	}

	private void calcPaths(){
		Graph<T, RepastEdge<T>> graph = null;
		if (net instanceof JungNetwork)
			graph = ((JungNetwork)net).getGraph();
		else if (net instanceof ContextJungNetwork)
			graph = ((ContextJungNetwork)net).getGraph();
		RepastEdge myEdge = graph.getEdges().iterator().next();
		dsp = new DijkstraShortestPath<T,RepastEdge<T>>(graph, transformer);
	}

	public void projectionEventOccurred(ProjectionEvent<T> evt) {
		if (evt.getType() != ProjectionEvent.OBJECT_MOVED) {
			calc = true;
		}
	}

	public void finalize() {
		if (net != null)
			net.removeProjectionListener(this);
	}
}
```

D.4 Vehicular communication

D.4.1 CommunicationMessage.java

```
public class CommunicationMessage {

	private double x;
	private double y;
	private String name;
	private String vehicleType;
	private double speed;
	private boolean stopped;
	private Junction previousJunction;
	private Junction nextJunction;
	private ArrayList<Junction> comingJunctions;
private RoadEdge currentRoadEdge;
private RoadEdge nextRoadEdge;
private double remainingDistanceToNextJunction;
private double remainingTimeToNextJunction;
private boolean bCloserToEmergencyVehicleInRange;
private double prevDistanceToEmergencyVehicleInRange;
private double closestDistnaceToEmergencyVehicleInRange;
private int preferedLane;

public CommunicationMessage() {
	this.previousJunction = null;
	this.nextJunction = null;
	this.comingJunctions = new ArrayList<Junction> ();
	this.currentRoadEdge = null;
	this.nextRoadEdge = null;
	this.speed = 0.0;
	this.remainingDistanceToNextJunction = 0.0;
	this.remainingTimeToNextJunction = 0.0;
	this.x = 0.0;
	this.y = 0.0;
}

public double getX() {
	return x;
}
public void setX(double x) {
	this.x = x;
}
public double getY() {
	return y;
}
public void setY(double y) {
	this.y = y;
}
public boolean isAlreadyUpdated(double myx, double myy) {
	boolean result = false;
	if (Double.compare(this.x, myx) == 0 && Double.compare(this.y, myy) == 0 )
										result = true;
	else result =  false;
	return result;
}

public boolean isStopped() {
	return stopped;
}

public void setStopped(boolean stopped) {
	this.stopped = stopped;
}

public String getName() {
	return name;
}

public void setName(String name) {
	this.name = name;
}
```

```
public String getVehicleType() {
		return vehicleType;
}

public void setVehicleType(String vehicleType) {
		this.vehicleType = vehicleType;
}

public double getDiffSpeed(double distanceDiff) {
		if(distanceDiff < 0 && this.stopped) return 0.0;
		else if (distanceDiff < -20.1168 && !this.stopped) return 0.0;
		else return speed;
}

public double getSpeed() {
		return speed;
}

public void setSpeed(double speed) {
		this.speed = speed;
}

public int getPreferedLane() {
		return preferedLane;
}

public void setPreferedLane(int preferedLane) {
		this.preferedLane = preferedLane;
}

public Junction getPreviousJunction() {
		return previousJunction;
}

public void setPreviousJunction(Junction previousJunction) {
		this.previousJunction = previousJunction;
}

public Junction getNextJunction() {
		return nextJunction;
}

public void setNextJunction(Junction nextJunction) {
		this.nextJunction = nextJunction;
}

public ArrayList<Junction> getComingJunctions() {
		return comingJunctions;
}

public void setComingJunctions(ArrayList<Junction> comingJunctions) {
		this.comingJunctions = comingJunctions;
}

public void addComingJunction(Junction comingJunction) {
		this.comingJunctions.add(comingJunction);
```

```
}

public void clearComingJunctions () {
	this.comingJunctions.clear();
}

public RoadEdge getCurrentRoadEdge() {
	return currentRoadEdge;
}

public void setCurrentRoadEdge(RoadEdge currentRoadEdge) {
	this.currentRoadEdge = currentRoadEdge;
}

public RoadEdge getNextRoadEdge() {
	return nextRoadEdge;
}

public void setNextRoadEdge(RoadEdge nextRoadEdge) {
	this.nextRoadEdge = nextRoadEdge;
}

public double getRemainingDistanceToNextJunction() {
	return remainingDistanceToNextJunction;
}

public void setRemainingDistanceToNextJunction(
			double remainingDistanceToNextJunction) {
	this.remainingDistanceToNextJunction = remainingDistanceToNextJunction;
}

public double getRemainingTimeToNextJunction() {
	return remainingTimeToNextJunction;
}

public void setRemainingTimeToNextJunction(double remainingTimeToNextJunction) {
	this.remainingTimeToNextJunction = remainingTimeToNextJunction;
}

public double getRemainingTimeToBreakdown(CommunicationMessage breakdownOnt) {
	double remainingDistance = this.getRemainingDistanceToBreakdown(breakdownOnt);
	double remainingTime = remainingDistance / (this.getSpeed()
					* GlobalVariables.ConversionCoefficient4MeterPerSec);
	return remainingTime; //sec
}

public double getRemainingDistanceToTheVehicleBehind(
						CommunicationMessage behindVehicleOnt) {
	return behindVehicleOnt.getRemainingDistanceToTheVehicleAhead(this);
}

public double getRemainingDistanceToBreakdown(CommunicationMessage breakdownOnt) {
	return this.getRemainingDistanceToFrontVehicle(breakdownOnt);
}

public double getRemainingDistanceToTheVehicleAhead(
						CommunicationMessage frontVehicleOnt) {
        return this.getRemainingDistanceToFrontVehicle(frontVehicleOnt);
}

public double getRemainingDistanceToFrontVehicle(
                                        CommunicationMessage frontVehicleOnt) {
        boolean bOnTheSameRoad = false;
        boolean bComingToTheRoad = false;
        double distance = -1.0;
        if (this.getCurrentRoadEdge().getName().equals(
                frontVehicleOnt.getCurrentRoadEdge().getName())) bOnTheSameRoad = true;
        if (this.nextRoadEdge != null && this.getNextRoadEdge().getName()
                .equals(frontVehicleOnt.getCurrentRoadEdge().getName()) &&
                this.getNextJunction().getName().equals(frontVehicleOnt
                        .getPreviousJunction().getName())) bComingToTheRoad = true;
        if (bOnTheSameRoad) {
                distance =  this.getRemainingDistanceToNextJunction()
                                - frontVehicleOnt.getRemainingDistanceToNextJunction();
        } else if (bComingToTheRoad) {
                distance = this.getRemainingDistanceToNextJunction()
                + (frontVehicleOnt.currentRoadEdge.getRoadElement().getShape_Leng()
                - frontVehicleOnt.getRemainingDistanceToNextJunction());
        }
        if (distance > Vehicle.vlength) return distance - Vehicle.vlength;
        else if (distance > 0 && distance <= Vehicle.vlength) return Vehicle.vlength / 2.0;
        return distance;
}

public boolean isInTheSituationOfVehicleStationary(
                CommunicationMessage breakdownOnt, double givenDistanceRange) {
        boolean bCase1, bCase2;
        bCase1 = false;
        bCase2 = false;
        boolean bOnTheSameRoad = false;
        boolean bComingToTheRoad = false;
        double distance = givenDistanceRange * 2;
try {
        if (this.getCurrentRoadEdge().equals(breakdownOnt.getCurrentRoadEdge()))
                                                        bOnTheSameRoad = true;
        if (this.nextRoadEdge != null &&
                this.getNextRoadEdge().equals(breakdownOnt.getCurrentRoadEdge()) &&
                this.getNextJunction().equals(breakdownOnt.getPreviousJunction()))
                        bComingToTheRoad = true;
        if (bOnTheSameRoad) {
                distance =  this.getRemainingDistanceToNextJunction()
                        - breakdownOnt.getRemainingDistanceToNextJunction();
                if (distance > 0 && distance < givenDistanceRange) bCase1 = true;
        } else if (bComingToTheRoad) {
                distance = this.getRemainingDistanceToNextJunction()
                + (breakdownOnt.currentRoadEdge.getRoadElement().getShape_Leng()
                - breakdownOnt.getRemainingDistanceToNextJunction());
                if (distance < givenDistanceRange) bCase2 = true;
        }
} catch (Exception e) {

        e.printStackTrace();
        return false;
}
```

```
                return bCase1 || bCase2;
        }

        public boolean updateRelativeLocation(Vehicle thisV) {
                try {
                        Coordinate currentCoord
                        = thisV.infra.getRoadGeography().getGeometry(thisV).getCoordinate();
                        Point currentPoint = new GeometryFactory().createPoint(currentCoord);
                        double minDist = Double.MAX_VALUE;
                        RoadEdge nearestRoadEdge = null;
                        if (this.getName() == null) {
                                this.setName(thisV.getName());
                                this.setVehicleType(thisV.getVehicleType());
                        }
                        this.setX(currentPoint.getX());
                        this.setY(currentPoint.getY());
                        this.setSpeed(thisV.speed);
                        this.setPreferedLane(thisV.getPreferedLane());
                        boolean flag = false;
                        RoadEdge roadEdge = null;
                        RoadElement roadElement = null;
                        double thisDist = minDist;
                        for (int i =0; i < thisV.route.getRouteEdges().size(); i++) {
                                roadEdge = thisV.route.getRouteEdges().get(i);
                                roadElement = roadEdge.getRoadElement();
                                thisDist = currentPoint.distance(thisV.infra
                                        .getRoadGeography().getGeometry(roadElement));
                                if (thisDist < minDist) {
                                        minDist = thisDist;
                                        nearestRoadEdge = roadEdge;
                                } // if thisDist < minDist
                        }
                        if (nearestRoadEdge == null)
                        this.setCurrentRoadEdge(nearestRoadEdge);
                        int currentIndex = thisV.route.getRouteEdges().indexOf(nearestRoadEdge);
                        if (currentIndex < thisV.route.getRouteEdges().size() - 1)
                           this.setNextRoadEdge(thisV.route.getRouteEdges().get(currentIndex + 1));
                        else this.setNextRoadEdge(null);
                        if (this.getCurrentRoadEdge() == null) return flag;
                        if (this.getCurrentRoadEdge().isFlipped()) {
                           this.setPreviousJunction(nearestRoadEdge.getRoadElement().toJunc);
                           this.setNextJunction(nearestRoadEdge.getRoadElement().fromJunc);
                        } else {
                           this.setPreviousJunction(nearestRoadEdge.getRoadElement().fromJunc);
                           this.setNextJunction(nearestRoadEdge.getRoadElement().toJunc);
                        }
                        flag = true;
                        Coordinate fromCoord = currentCoord;
                        Coordinate toCoord;
                        double eachDistance, remainDistance = 0, remainTime;
                        for (int i=0; i < thisV.route.getRouteCoords().size(); i++) {
                                if (i > 0) fromCoord = thisV.route.getRouteCoords().get(i-1);
                                toCoord = thisV.route.getRouteCoords().get(i);
                                eachDistance = fromCoord.distance(toCoord);
                                remainDistance += eachDistance;
                                if(toCoord.equals(this.getNextJunction().getCoordinate())) break;
                        }
```

```
			this.setRemainingDistanceToNextJunction(remainDistance);
			remainTime = remainDistance / (thisV.getSpeed() * 1609.344 / 3600.0);
			this.setRemainingTimeToNextJunction(remainTime);
			return flag;
		} catch (Exception e) {
			e.printStackTrace();
		}
		return false;
	}

	public boolean isInTheSituationOfEmergencyVehicle (CommunicationMessage ambulOnt) {
		double givenDistanceRange = GlobalVariables.commrange; //200.0;
		double givenTimeRange = givenDistanceRange
			/ (GlobalVariables.TRAVEL_SPEED_MPH * 1.5 * 1609.344 / 3600.0);
		double givenTimeDiff = -1.0;
		double givenDistDiff = givenTimeDiff
			* (GlobalVariables.TRAVEL_SPEED_MPH * 1.5 * 1609.344 / 3600.0);
		boolean bCase1= false, bCase2 = false, bCase3 = false;
		if (this.currentRoadEdge == null) return false;
	  try {
		double distanceDiff = -100.0, timeDiff = -10.0;
		//case1 on the same road
		if (this.currentRoadEdge.getName().equals(
						ambulOnt.getCurrentRoadEdge().getName())) {
			distanceDiff = ambulOnt.getRemainingDistanceToNextJunction()
						- this.getRemainingDistanceToNextJunction();
			timeDiff = distanceDiff / ((ambulOnt.getSpeed()
					- this.getDiffSpeed(distanceDiff) ) * 1609.344 / 3600.0);
			if ( (distanceDiff < givenDistanceRange && distanceDiff > givenDistDiff)   ||
			  (timeDiff < givenTimeRange && timeDiff > givenTimeDiff)) {
				bCase1 = true;
			}
		}
		//case1 on the same road, but opposite lane
		else if (this.currentRoadEdge.getRoadElementName().equals(
				ambulOnt.getCurrentRoadEdge().getRoadElementName())) {
			distanceDiff = ambulOnt.getRemainingDistanceToNextJunction()
				+ this.getRemainingDistanceToNextJunction()
				- this.currentRoadEdge.getRoadElement().getShape_Leng();
			timeDiff = distanceDiff / ((ambulOnt.getSpeed()
				+ this.getDiffSpeed(distanceDiff)) * 1609.344 / 3600.0);
			if ( (distanceDiff < givenDistanceRange && distanceDiff > givenDistDiff)
				|| (timeDiff < givenTimeRange && timeDiff > givenTimeDiff)) {
						bCase1 = true;
					}
		}
		//case 2 not on the same road, v1 is not on the road that a1 is heading to,
		//but they are heading to the same junction
		else if (ambulOnt.getNextRoadEdge() != null && this.getNextJunction() !=null
			&& ambulOnt.getNextJunction() != null
			&& !this.currentRoadEdge.getRoadElementName().equals(
				ambulOnt.getNextRoadEdge().getRoadElementName())
			&& this.getNextJunction().equals(ambulOnt.getNextJunction())) {
			distanceDiff = this.remainingDistanceToNextJunction
				+ ambulOnt.getRemainingDistanceToNextJunction();
			timeDiff = distanceDiff / ((ambulOnt.getSpeed()
				+ this.getDiffSpeed(distanceDiff)) * 1609.344 / 3600.0);
			if ( (distanceDiff < givenDistanceRange && distanceDiff > givenDistDiff)
					|| (timeDiff < givenTimeRange && timeDiff > givenTimeDiff)) {
					bCase2 = true;
			}
		}
		//case 3 v1 is on the road that a1 is heading to
		else if (ambulOnt.getNextRoadEdge() != null &&
		   this.currentRoadEdge.getName().equals(ambulOnt.getNextRoadEdge().getName())){
			distanceDiff = this.currentRoadEdge.getRoadElement().getShape_Leng()
					- this.getRemainingDistanceToNextJunction()
					 + ambulOnt.getRemainingDistanceToNextJunction(); ;
			timeDiff = distanceDiff / ((ambulOnt.getSpeed()
					- this.getDiffSpeed(distanceDiff)) * 1609.344 / 3600.0);
			if ( (distanceDiff < givenDistanceRange && distanceDiff > givenDistDiff)   ||
					  (timeDiff < givenTimeRange && timeDiff > givenTimeDiff)) {
					bCase3 = true;
			}
		}
		//case 3 v1 is on the road that a1 is heading to, opposite lane
		else if (ambulOnt.getNextRoadEdge() != null && this.currentRoadEdge
	.getRoadElementName().equals(ambulOnt.getNextRoadEdge().getRoadElementName())) {
			distanceDiff = this.getRemainingDistanceToNextJunction()
						+ ambulOnt.getRemainingDistanceToNextJunction();
			timeDiff =  distanceDiff / ((ambulOnt.getSpeed()
						+ this.getDiffSpeed(distanceDiff)) * 1609.344 / 3600.0);
			if ( (distanceDiff < givenDistanceRange && distanceDiff > givenDistDiff)   ||
					 (timeDiff < givenTimeRange && timeDiff > givenTimeDiff)) {
					bCase3 = true;
			}
		}
		//case 4 a1 is already passed the v1 and a1 is on the road that v1 is heading to,
		//but their distance is closer than the given range
		else if (this.nextRoadEdge != null && this.stopped && ambulOnt.currentRoadEdge
							.getName().equals(this.nextRoadEdge.getName())){
			distanceDiff = -1.0
			* (ambulOnt.currentRoadEdge.getRoadElement().getShape_Leng()
			- ambulOnt.getRemainingDistanceToNextJunction()
			 + this.getRemainingDistanceToNextJunction());
			timeDiff = distanceDiff / ((ambulOnt.getSpeed()
					- this.getDiffSpeed(distanceDiff)) * 1609.344 / 3600.0);
		if ( (distanceDiff < givenDistanceRange && distanceDiff > givenDistDiff)   ||
				 (timeDiff < givenTimeRange && timeDiff > givenTimeDiff)) {
					bCase3 = true;
			}
		}
		//case 4 a1 is already passed the v1 and a1 is on the road that v1 is heading to,
		//but their distance is closer than the given range
		//opposite lane
		else if (ambulOnt.getNextRoadEdge() != null && this.stopped
			&&  this.previousJunction.equals(ambulOnt.getPreviousJunction())
			&& !this.currentRoadEdge.getRoadElementName().equals(
						ambulOnt.getNextRoadEdge().getRoadElementName())) {
			distanceDiff = -1.0
					* (this.currentRoadEdge.getRoadElement().getShape_Leng()
					- this.getRemainingDistanceToNextJunction()
					+ ambulOnt.currentRoadEdge.getRoadElement().getShape_Leng()
					- ambulOnt.getRemainingDistanceToNextJunction());
```

```
                    timeDiff =  distanceDiff / ((ambulOnt.getSpeed()
                              + this.getDiffSpeed(distanceDiff)) * 1609.344 / 3600.0);
                    if ( (distanceDiff < givenDistanceRange && distanceDiff > givenDistDiff)
                              || (timeDiff < givenTimeRange && timeDiff > givenTimeDiff)) {
                              bCase3 = true;
                    }
          }
          else {
                    distanceDiff = this.getRemainingDistanceToNextJunction()
                                        + ambulOnt.getRemainingDistanceToNextJunction();
          }
    } catch (Exception e) {
          e.printStackTrace();
    }
          return bCase1 || bCase2 || bCase3;
}

public String toString () {
  String str = "name: " + this.name + "\n" + "";
 str = str + this.name + "-" + "previousJunction: " + this.previousJunction.getName() + "\n" + "";
  str = str + this.name + "-" +"nextJunction: " + this.nextJunction.getName() + "\n" + "";
  str = str + this.name + "-" +"currentRoadEdge: "
                    + this.currentRoadEdge.getRoadElement().getFID() + "\n" + "";
  str = str + this.name + "-" +"nextRoadEdge: "
                    + this.nextRoadEdge.getRoadElement().getFID() + "\n" + "";
  str = str + this.name + "-" +"remainingDistanceToNextJunction: "
                    + this.remainingDistanceToNextJunction + "\n" + "";
  str = str + this.name + "-" +"remainingTimeToNextJunction: "
                    + this.remainingTimeToNextJunction +  "\n" + "";
  return str;
}
}
```