

DOCTOR OF PHILOSOPHY

Semantic agent-based controls in service oriented architecture (SOA) enabled intelligent transportation systems (ITS)

Kamran, Shoaib

Award date:
2011

Awarding institution:
Coventry University

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of this thesis for personal non-commercial research or study
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission from the copyright holder(s)
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

SEMANTIC AGENT-BASED CONTROLS IN SERVICE ORIENTED ARCHITECTURE (SOA) ENABLED INTELLIGENT TRANSPORTATION SYSTEMS (ITS)

SHOAIB KAMRAN

September 2011



A thesis submitted in partial fulfilment of the University's requirements for the
Degree of Doctor of Philosophy

SEMANTIC AGENT-BASED CONTROLS IN SERVICE ORIENTED ARCHITECTURE (SOA) ENABLED INTELLIGENT TRANSPORTATION SYSTEMS (ITS)

SHOAIB KAMRAN

September 2011



A thesis submitted in partial fulfilment of the University's requirements for the
Degree of Doctor of Philosophy

*Semantic Agent-based Controls in Service Oriented Architecture (SOA)
enabled Intelligent Transportation Systems (ITS)*



Abstract

Intelligent Transportation Systems (ITS) have been considered as a solution to the problems associated with modern urban traffic management systems. The implementation of ITS is highly complex due to the multi-domain and geographically distributed nature of the traffic controls/systems and legacy infrastructures. Some of the approaches that have attempted to address these challenges include control engineering, artificial intelligence, telematics, grid computing and multi-agent systems. This research work was undertaken to investigate the implementation approaches aimed at addressing the complexity and communication challenges of ITS, specifically intelligent and cooperative traffic systems/controls which are capable of distributed decision making in response to emergent traffic situations.

This research introduces the novel concept of Semantic Agent-based Controls which utilise a unique combination of computing concepts including multi-agent systems, Service Oriented Architecture (SOA) and semantic web services. The semantic agent-based controls extend the functionality and decision making capabilities of fixed traffic controls by using semantic agents over the network to effectively control geographically distributed traffic controls/roadside systems.

The research introduces a novel Agent communication and coordination mechanism that facilitates the interaction between the Semantic Agent-based Controls in a multi-domain ITS environment. The mechanism uses a semantic layer approach based on the principles of SOA and the concept of semantic web services which enables Agents to store, share and integrate heterogeneous data on the semantic level across the SOA enabled network of traffic systems and devices. This highly flexible mechanism allows semantic agent-based controls to interact with each other by using different levels of ontologies that semantically describe the Agents' intentions, behaviours and coordination strategies. The semantic communication layer allows coordination of action plans in the form of decision rules between the semantic agent-based controls which facilitates dynamic and distributed decision making in response to emergent situations.

A new ITS platform (named ITS@CU) is presented, which was developed to support and evaluate the semantic agent-based controls approach. The ITS@CU platform (and the associated simulation utilities) was developed using the latest state of the art commercially available tools and technologies. It is a multi-agent platform and is based on advanced SOA principles which enable seamless integration of traffic controls/systems.

The research approaches are evaluated using a variety of different test cases generated using historical traffic data and typical traffic situations identified by UTM Control Room, Coventry City Council. The results demonstrate the advantages of Semantic Agent-based Controls especially when compared with traditional controls (without Agents). The advantages include communication reductions, flexible and robust coordination between the distributed controls, and enhanced decision making and self-organisation capabilities.

The research presented started as a Knowledge Transfer Programme (KTP) funded project in collaboration with T@lecom Limited. Elements of the ITS@CU platform developed during the project also led to the design and development of the following three successful commercial applications/products for T@lecom. The “Patient Transport System” is a mobile solution for ambulance crews to transfer patients to and from hospitals and with novel features of intelligent tracking, two-way messaging/alerts and patient prioritising in real-time. The “Mobile Gateway System” is a SOA enabled unified communication and integration platform for T@lecom’s mobile solutions. Finally, the “Vehicle Tracking Solution” incorporated comprehensive functionalities of the ITS@CU platform and also includes a novel driver behaviour assessment.

Keywords: Agent-based Controls, SOA, ITS, Multi-Agent Systems, Grid Computing, Telematics, Urban Traffic Management and Semantic Web Services.

Acknowledgement

I am deeply grateful to my supervisor, Dr. Olivier Haas for his efforts, inspiring suggestions and on-going help and guidance for my research work. I would also like to thank Prof. Keith Burnham for providing great level of support and several CTAC students who contributed to the research in different ways.

Finally, I would also like to thank my colleagues at T@lecom limited for their help and providing resources for this research.

Table of Contents

Abstract	3
Acknowledgement.....	5

1. Research Introduction 11

1.1. Context and Problem Statement.....	12
1.2. Research Aim & Objectives.....	14
1.3. Research Methodology	16
1.4. Research Contributions	21
1.5. Scope of the Report.....	23
1.6. Report Organisation	23

2. Background & Literature Review 26

2.1. Intelligent Transportation Systems (ITS)	27
2.1.1. ITS overview.....	27
2.1.2. ITS controls and traffic data management.....	29
2.1.3. Traffic light controls	32
2.1.4. Traffic flow simulation.....	33
2.1.5. Traffic network modelling and mapping.....	34
2.1.6. Incident management	36
2.1.7. ITS in the context of this research.....	37
2.2. Agent Oriented Systems and Controls	39
2.2.1. Agent technology overview	39
2.2.2. Multi-Agent System (MAS)	41
2.2.3. Multi-Agent communication and coordination.....	42
2.2.4. Security in agent-oriented environment.....	44
2.2.5. Virtual Agent organisations	45
2.2.6. Agent oriented design approaches.....	46
2.2.7. Agent tools and technologies.....	52
2.2.8. Agent-oriented approach to ITS problems.....	53
2.2.9. Difference between Agents and other technologies	54
2.2.10. Critical analysis and challenges of agents based approach.....	54
2.3. Ontologies for Agent communication.....	56
2.3.1. Ontology overview	56
2.3.2. Ontologies design.....	56
2.3.3. Ontology languages and technologies review.....	57
2.4. Semantic Agent-based Controls & Services	58
2.4.1. Agent-based Controls.....	58
2.4.2. Semantic web services	58

2.4.3.	Semantic Agent	59
2.5.	Service Oriented Architecture and Grid computing	60
2.5.1.	SOA Overview	60
2.5.2.	SOA principles and architecture	62
2.5.3.	SOA related technologies	65
2.5.4.	Critical analysis and drawbacks of SOA	66
2.5.5.	SOA in the context of this research	67
2.5.6.	Grid computing	68
2.6.	Review of the Related Work	70
2.6.1.	Related research platforms and systems	70
2.6.2.	Relevant Agent oriented approaches and studies	73
2.6.3.	Related commercial traffic management systems	77
2.7.	Conclusion	79
3.	Road Network Model Development	80
3.1.	Overview	81
3.2.	Road network elements	81
3.3.	Road network mapping	82
3.4.	Model description	83
3.5.	Conclusion	88
4.	Agent-based Controls Design & Organisation Structure	89
4.1.	Agents design description	90
4.1.1.	Types of Agents	90
4.1.2.	Control Agents	91
4.1.3.	Service Agents	101
4.1.4.	Operational Agents	108
4.2.	Agents organisational structure	116
4.3.	Conclusion	118
5.	Agent Communication Structure	119
5.1.	Multi-agent communication layer	120
5.2.	Agent message structure	122
5.3.	Agent communication lifecycle	128
5.4.	Security	130
5.5.	Conclusion	131

6. Agent Semantics, Ontologies and Rules structure 132

6.1.	Agent message flow	133
6.2.	Semantic Layer (Semantic-Content) description	135
6.3.	Ontology structure.....	137
6.4.	Rules structure.....	143
6.5.	Agent Plans.....	144
6.6.	Conclusion	145

7. Implementation of ITS@CU platform..... 146

7.1.	Platform overview	147
7.2.	Design and architecture	148
7.2.1.	Platform components.....	148
7.2.2.	Platform components design architecture	149
7.2.3.	SOA reference architecture of the platform	153
7.3.	Components development and design description	155
7.3.1.	Central Control system.....	155
7.3.1.1.	Infrastructure description.....	155
7.3.1.2.	Services on the central control system	157
7.3.1.3.	Applications in the central control system.....	160
7.3.1.4.	Central Database/Knowledge base of the platform.....	164
7.3.2.	Traffic infrastructure/Grid Controls.....	166
7.3.2.1.	Grid Controller Application.....	166
7.3.2.2.	Agent Communication Interface/Layer	167
7.3.2.3.	Grid Database/Knowledge base	168
7.3.2.4.	Traffic Controllers applications.....	169
7.3.3.	Vehicle Control System.....	171
7.4.	Mobile Application Development Framework (MADF)	173
7.5.	Conclusion	176

8. Evaluation & Analysis..... 177

8.1.	Evaluation approach and simulation setup.....	178
8.1.1.	Study area overview	178
8.1.2.	Simulation setup overview	179
8.1.3.	Platform and Agents configuration	180
8.1.4.	Test data.....	188
8.2.	Study cases and result analysis.....	189
8.2.1.	Problem detection & response (Incident scenario)	189
8.2.2.	Communication approach evaluation.....	205
8.2.3.	Semantic web services approach	212

8.3.	General observations & discussion	214
8.3.1.	Reliability.....	214
8.3.2.	Security handling	216
8.3.3.	Centralisation and decentralisation flexibility	217
8.3.4.	Maintenance and scalability	217
8.3.5.	Limitations.....	218
8.4.	Conclusion and summary of findings.....	220

9. Conclusion & Further Work 222

9.1.	Conclusion.....	223
9.2.	Other potential application areas (Non-ITS).....	228
9.3.	Further Work	229

References	231
-------------------------	------------

Appendices

Appendix A:	Commercial implementations & contributions
Appendix B:	Other research implementations & contributions
Appendix C:	Intelligent Transportation Systems (ITS) Review
Appendix D:	Agent Oriented Technologies Review
Appendix E:	Ontology languages overview
Appendix F:	Review of the Tools & Technologies relevant to this research
Appendix G:	Wireless Communication Technologies Review
Appendix H:	ITS@CU Platform Components design details
Appendix I:	Mobile Application Development Framework (MADF) Description
Appendix J:	CD Contents and project Source Code
Appendix K:	NetLogo simulation study for traffic evaluation
Appendix L:	SCOOT data and traffic area study report
Appendix M:	Schema XSD – Agent Message Structure
Appendix N:	Schema XSD – Semantic Content Structure
Appendix O:	ITS@CU Ontology Structure
Appendix P:	ITS@CU Applications Screenshots

Abbreviations

ACC	Agent Communication Channel
ACL	Agent Communication Language
AMS	Agent Management System
API	Application Programming Interface
BDI	Belief-Desire-Intention
CORBA	Common Object Request Broker Architecture
CTAC	Control Theory and Application Centre
DBMS	Database Management System
DTD	Document Type Definition
FIPA	Foundation for Intelligent Physical Agents
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IDE	Integrated Development Environment
IrDA	Infrared Data Association
ITS	Intelligent Transportation Systems
JADE	Java Agent DEvelopment Framework
LINQ	Language Integrated Query
MADF	Mobile Application Development Framework
OMG	Object Management Group
OODBS	Object-Oriented Database Systems
P2P	Peer-to-Peer
RDF	Resource Description Framework
REST	Representational State Transfer
SDK	Software Development Kit
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifier
UTMC	Urban Traffic Management & Control
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XHTML	eXtensible Hyper Text Markup Language
XML	Extensible Markup Language
XSD	XML Schema Definition

Chapter 1

Research Introduction

This chapter introduces this research and presents its overall aim and objectives. It also describes the research methodology used and the organisation of the thesis. Additionally, the chapter outlines some of the key research and commercial contributions, published research papers and novelty aspects of this research.

1.1. Context and Problem Statement

Intelligent Transportation Systems (ITS) have been considered as an alternative approach with a potential to revolutionise our modern traffic infrastructure and resolve traffic management problems. During the last 5 years, advanced ITS-based traffic infrastructures are gradually becoming a reality due to the emergence of high-tech computing technologies and the integration capabilities of the new generation of traffic systems and devices. However, the implementation of ITS traffic infrastructures still faces a number of challenges due to the highly complex, multi-domain, heterogeneous, and distributed nature of the data, systems and devices involved in ITS infrastructures. Other obstacles include a lack of standards, high processing power/memory requirements within traffic devices, multiple decision levels, data load and network latency (Wang, 2005). These result in a lack of efficient communication and integration between the various ITS components/systems, which is vital for the efficient management of traffic flow, incident detection and for a quicker incident response time. It also makes the infrastructure design, system development, implementation and maintenance of ITS difficult as well as costly. Obviously, the bigger the ITS infrastructure grows the more complex it becomes to manage especially when it comes to incident/fault detection within ITS systems, and managing a quick and efficient response to it.

In order to address these issues and problems, different approaches have been adopted by other research, ranging from artificial intelligence, multi-Agent networks, artificial neural networks, distributed systems, telematics to grid Computing. So far, the outcome of these research approaches and systems have not been able to effectively address some of the important ITS problems, such as:

- How to establish cooperation, coordination and communication among different domains and systems
- How to store, share and integrate heterogeneous data on the semantic level across the network of traffic systems and devices
- Finding an intelligent way to control the geographically distributed and limited memory roadside sensors/systems
- How to manage dynamic service flow and self-organisation of the traffic systems in response to emergent situations

In comparison to large scale corporate and commercial systems, ITS related systems have still not fully benefited from technologies such as the new generation SOA, federated Cloud computing and latest ad-hoc wireless technologies. Utilising intra/internet enabled services based infrastructure (in a controlled manner) can beneficially provide more functionality to ITS controls without additional overheads.

Among several other technologies, researchers considered “Agent-based technologies” as a potential and well-suited approach for such complex ITS environments (Wooldridge, 2002; Wang, 2005). However, Agent-based

technology has its limitations and are lacking in commercial tools/technologies, so it has been unable to address all these issues effectively on its own. Therefore, the solution to such complex problems requires a fundamental shift in approach.

This research has approached current ITS issues from a different perspective by utilising a combination of computing concepts such as multi-agent systems, service oriented architecture (SOA), semantic web services and grid computing in order to implement an ITS platform (named ITS@CU in this research). The implementation was based on latest state of the art and commercially available tools and technologies. SOA was adopted for the seamless integration and communication among different traffic control systems from multiple domains. Semantic Agent-based controls were used to control geographically distributed traffic controls/systems and provide additional functionality and decision making capabilities. A semantic layer (based on semantic web services) was introduced to all agent and service communication with different levels of ontologies. The semantic layer enables agent controls to store, share and integrate heterogeneous data on the semantic level across the network of traffic systems and devices. SOA principles were used for providing composite services (multiple service work flows) and the ability to dynamically discover services across the service bus. This facilitated the traffic systems self-organisation in response to emergent situations using dynamic service flows.

This PhD research was initially started *as part* of a Knowledge Transfer Programme (KTP)¹ project at T@lecom Limited² in association with the Control Theory and Applications Centre (CTAC), Coventry University. T@lecom is one of the leading providers of transportation/logistics related mobile software solutions, and this research project was primarily targeted towards their R&D efforts to implement a new platform for the development of a new generation of ITS-based systems at T@lecom. The KTP project successfully completed within its two year term period (2005-07), however the author continued the research project at T@lecom (during his employment for the next two years), and the PhD at CTAC. The objectives of the PhD were derived from the KTP/T@lecom research requirements however its research scope goes much further to another level in the subject research areas.

¹ **Knowledge Transfer Partnerships (KTP)** is Europe's leading programme helping businesses to improve their competitiveness and productivity through the better use of knowledge, technology and skills that reside within the UK knowledge base. It supports and funds a huge range of projects in joint collaboration between commercial organisations and universities/research organisations.

² **T@lecom Limited** is a Microsoft's Gold Partner specialises in enterprise mobility solutions such as mobile application for field/sales force automation, logistics, vehicle tracking, navigation assistance, ambulance and emergency response systems, and enterprise Blackberry solutions. The main sector serviced by T@lecom is the NHS Ambulance Services and their Patient Transportation System (PTS), and currently T@lecom is the most dominant player in that sector.

1.2. Research Aim & Objectives

This research investigated novel methods and implementation approaches aimed at addressing the complexity and communication challenges of ITS, in particular, intelligent and cooperative traffic systems/controls which are capable of distributed decision making in response to emergent traffic situations in a complex multi-domain environment.

The following objectives were set to accomplish the overall aim of the research:

- Develop a road network model adapted for the ITS@CU³ platform
- Design and implement Semantic Agent-based Controls to effectively control the distributed ITS traffic systems/controls and organise the Agents in an effective organisational structure.
- Design and implement a novel Agent-based communication and cooperation mechanism between the ITS traffic systems/controls and services distributed over the grids of the road network. The communication mechanism uses SOA principles for the seamless systems/controls integration, efficient service discovery and dynamic service composition. The mechanism also uses Semantic web service concepts allowing the Agents to use different levels of dynamic ontologies.
- Define Ontologies to semantically describe the Agents' behaviours, rules and coordination strategies for flexible and efficient communications, and distributed decision making in the multi-agent environment of the platform.
- Design and develop the ITS@CU platform (and associated simulation utilities) using latest state of the art commercially available tools and technologies. The platform provides and supports various layers of services in order to manage, integrate and control the distributed Agent-based Controls, and also facilitate the arbitration and decision making process between the Agents.
- Develop a new mobile application framework for developing PDA based applications to simulate the behaviour of different types of traffic Controls/devices (for platform evaluation)
- Evaluate and analyse the platform using various traffic scenarios and other related test cases.

Additionally, provide the commercial implementation benefits of the platform components in order to fulfil the KTP requirements.

³ **ITS@CU** – Intelligent Transportation System at Coventry University is only a research title to refer to the ITS platform developed as part of the KTP project.

The following *figure 1.1* outlines the overall aims and objectives of the research.

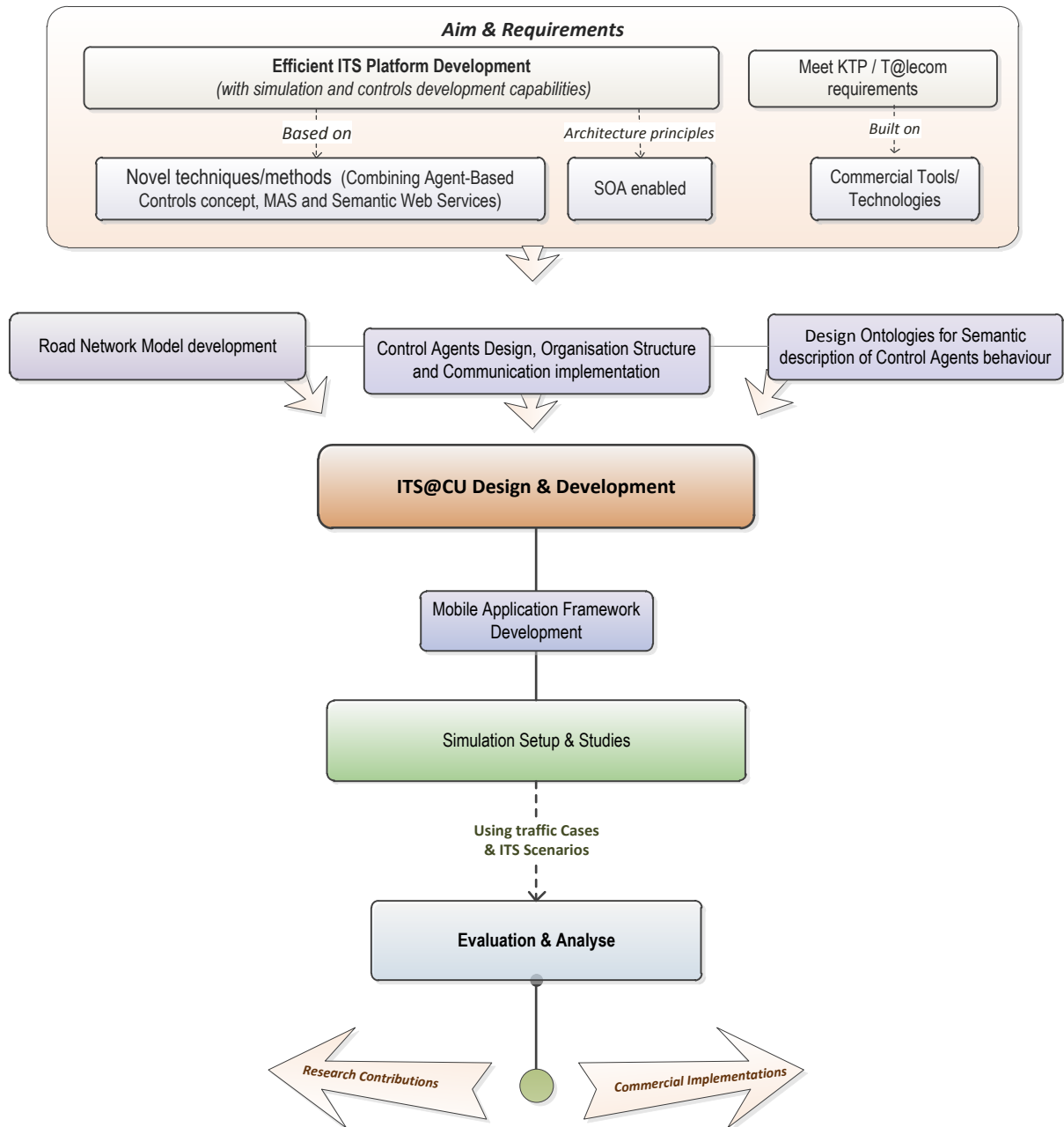


Figure 1.1: Overall Aim and objectives of the research

1.3. Research Methodology

This research has included various experiments, study analyses, field surveys, design & modelling, development, simulations and continual refinements. The author has liaised with the UTMC Control Room of Coventry City Council responsible for managing the traffic of Coventry City and surrounding areas. For the purpose of this research, the UTMC provided an extensive range of traffic data for different traffic routes in Coventry City, gathered from the SCOOT system (Siemens Mobility, 2009).

This research was a commercially-oriented R&D project and was therefore conducted in line with T@lecom's project management and software development processes using IBM's Rational Unified Processes (RUP). Various other methodologies were also considered, specifically Agile based, such as SCRUM, Kanban, eXtreme Programming (XP) and Dynamic Systems Development (DSDM) which promotes iterative, collaborative, adaptable and requirements driven development processes. Each methodology has its own strengths and drawbacks depending on the application area and project type.

The following table outlines some of the relevant methodologies considered for this research project:

Methodology	Consideration aspects in the context of this research
PRINCE2 Projects in Controlled Environments (PRINCE)	<p>It is a highly structured methodology and primarily focused towards project management, control and organisation (Huijbers, et al., 2004). PRINCE2 provides a clear beginning, managed phases/stages and an end of a project throughout its lifecycle to ensure the successful completion (or termination) of a project <i>i.e.</i> on time, within budget and conforming to its requirements. PRINCE2 is a generic project management methodology and can be used for non IT projects as well. It is endorsed by the UK government as the project management standard for public projects (APMG, 2011).</p> <p>Some of the main reasons for not adopting PRINCE2 include:</p> <p>It is primarily a project management methodology rather than a development methodology (APMG, 2011). This research required a methodology tailored towards software development. PRINCE2 usage involves producing a number of specific stage and phase documentation (Lefevre, 2011) which would have created additional overhead for this research project.</p> <p>PRINCE2 was not used within T@lecom and it requires familiarity amongst the project team and stakeholders to be successful (Huijbers, et al., 2004;</p>

	Collaris & Dekker, 2010).
XP eXtreme Programming	<p>XP is an Agile process driven software engineering methodology. It focuses on customer requirements and to minimise the risks of rapid requirement changes during the project life cycle (Wells, 2009; Amatriain, 2008). It supports smaller collaborative teams, and it is fairly simple and lightweight compared to other Agile methodologies (Huijbers, et al., 2004).</p> <p>The lightweight nature and small team support was quite favourable for this research project, but as this project was not constantly changing from the requirements perspective, other methodologies provided additional benefits and a better approach when compared to XP. Additionally, XP was not used in T@lecom and would have required extra effort (time/training) to use it for the PhD project.</p>
SCRUM	<p>SCRUM is also an Agile method of project management and it is based on an iterative approach for product/solution development. It implements the work (backlog) in sprints/phases and a daily scrum meeting takes place to assess the backlog for the current sprint in order to steer the project in the right direction.</p> <p>The iterative sprint approach of SCRUM was suitable for this project i.e. by dividing the PhD tasks into several sprints along with timely sprint reviews with supervisor and project sponsors to track and control the research progress. However, the major difficulty in adopting SCRUM was that it was not used in T@lecom. Additionally, SCRUM is more focused on team management and team work (Collaris & Dekker, 2010), whereas this research work was primarily based on the author, with occasional interaction with T@lecom's development team.</p>
RUP Rational Unified Process	<p>RUP is a software design methodology by IBM. It is widely used in the US and parts of Europe (Huijbers, et al., 2004; Collaris & Dekker, 2010). The key strength of RUP is its versatile, modular, iterative and adaptable nature allowing the development of a wide variety of projects/products both software and non-software. It is a comprehensive methodology providing not just a process framework but also a set of guidelines, example templates and</p>

	<p>integration with various applications for risk and requirements management, planning, design (UML) and testing applications. The integration with such applications allows the methodology to be tightly coupled with the actual tools which are used for the project design, development and implementation.</p> <p>The modular and iterative nature of RUP provided an effective approach to distribute the PhD research into various phases and iterations (<i>as mentioned in table 1.2</i>). The iterations in each phase provided better control, planning and progress assessment not only for the author, but also for the supervisor and project sponsors (T@lecom and KTP).</p> <p>The possibility to adapt RUP for a single person team (Collaris & Dekker, 2010) meant that the author had the freedom to continue the project in both individual capacity and where required in a team. RUP promotes clear requirements analysis and testing after each iteration (Huijbers, et al., 2004), which was also helpful during the initial stages of the project where the project involved various experimentation and study analyses. It also helped with managing the requirement changes which resulted from the research experiments and literature review.</p> <p>One of the key drivers for adopting RUP as a development methodology was its use in T@lecom. For every methodology to be successful it is vital that it is supported and understood by the team (Huijbers, et al., 2004; Collaris & Dekker, 2010), and as RUP was already the preferred methodology used in T@lecom it was the most sensible way forward.</p> <p>Beside T@lecom's requirement & preference of using RUP, other main reasons for adopting it was its versatility, successful projects implementation (IBM, 2007; Hanssen, Westerheim, & Bjørnson, 2006), comprehensive documentations, SOA support (IBM, 2007), requirements driven & iterative approach (Sabine & Karlheinz; IBM, 2007), and also its capability to adapt towards research driven projects (Huijbers, et al., 2004).</p>
--	---

Table 1.1: Methodologies considered for the PhD

In accordance with the RUP methodology, the overall research project was broken into four phases and each phase comprised of several iterations. The following table outlines the phases and activities carried out during the lifecycle of this research:

Phase	Description	Research activities and Outputs	
Inception	Establishing key research objectives and its feasibility.	<ul style="list-style-type: none"> RDC1 document (PhD proposal) Business case document for T@lecom 	Multiple iterations in each phase
Elaboration	<p>Knowledge Acquisition, Domain understanding and research analysis</p> <p>Specifying requirements, system modelling and project planning</p>	<ul style="list-style-type: none"> Requirement specification document and project plan produced for T@lecom/KTP Literature review covering the analysis, and comparison of ITS-related researches, systems, models and methods. Surveys of Coventry city centre's road traffic network and traffic network model. Analysed SCOOT data provided by UTMC Control Room, Coventry City Council. 	
Construction	<p>System design, Experimentation, Development, Testing, Results Analysis</p> <p>Change management and continual system refinement</p>	<ul style="list-style-type: none"> Experiment and testing of various programming technologies, communication methods, and hardware such as mobile devices, GPS receivers, sensors and servers. Mobile application framework development Personal Area Network (PAN) based Agent-Agent communication system for simulations. ITS@CU platform design and development Testing and result analysis System/model verification, validation and improvement 	

Transition	System components deployment, formal documentation and feedback assessment	<ul style="list-style-type: none"> • Deployment of the system components • Assessment of the feedback from T@lecom's technical team and CTAC • Formal documentation of each iteration's outcome • Thesis write-up • Viva preparation • Implement changes and corrections recommended by PhD examiners 	
-------------------	--	---	--

Table 1.2: Research methodology (phases and iterations according to RUP)

1.4. Research Contributions

Novelty

This research project contributed to the research area and novel in the following ways:

- It introduced a concept of “Semantic Agent-based Controls”, which extends the concept of traditional Agents and fixed controls to another level. It enables controlling multiple ITS-based controls/systems over a network by using Control Agents with the capability to encapsulate semantic data. This allows for efficient communication, cooperation and coordination between agent-based controls, and provides the ability for agents to work in synergy with other agents with dynamic adaptability in order to manage the traffic grids controls and make smart decisions in response to emergent situations.
- Development of a new ITS platform (ITS@CU) based on novel methods by using a combination of different computing areas such as Multi-Agent systems, Control Systems, SOA and Semantic web services.
- One of the key elements of this research project is the use of SOA principles in the ITS@CU platform adapted to facilitate Agent interactions in the form of services over highly complex, distributed and multi-domain networks. This is a unique implementation of multi-agent systems and provides a robust and highly scalable ITS platform.
- It implemented a new Mobile Application Development Framework for rapid application development and simulation of ITS-based traffic Controls and Agents. The framework provides multiple wireless communication channels and methods (3G/GPRS, Wi-Fi, Bluetooth, Infrared, Web Services, TCP/UDP) and built-in external hardware support.
- The research was based on a commercial development approach and used state of the art technologies, at the time the work was implemented, such as .NET 4, WCF, SQL Server 2008 spatial data, LINQ, Silverlight, Bing Maps API, Windows Mobile 6.5 and various wireless communication technologies and libraries.

Commercial Implementations

This research was commercially inclined and in addition to the core research objectives, the R&D efforts also led to the development of various commercial applications and Proof of Concepts (PoC). Some of the relevant commercial implementations and patent ideas which resulted from this research are covered in “*Appendix A*”.

Research Publications

During the research, the author has published the following scientific papers based on various outcomes of the research:

- “Detecting traffic incidents based on traffic patterns and vehicle behaviours using GPS”, In Proceedings of the 18th International Conference on Systems Engineering (ISCE06), Coventry, United Kingdom, 5-7th September 2006, pp. 201 – 206 by Kamran, S., Black, J. and Haas, O. C. L. (2006).
- "A Multilevel Traffic Incidents Detection Approach: Identifying Traffic Patterns and Vehicle Behaviours using real-time GPS data", In proceedings of the Intelligent Vehicles Symposium, IEEE, Istanbul, Turkey, 13-15th June 2007, pp. 912 – 917 by Kamran, S. and Haas, O.C.L, (2007).
- “Emergency response time optimisation using real-time traffic information”, In proceedings of the 7th International Conference on Transport Systems Telematics (TST’07), Katowice-Ustroń, Poland, 17-19th October 2007 by Jaskulowski, M., Kamran, S. and Haas, O.C.L, (2007).
- “Semantic Agent-based Controls for SOA enabled ITS”, In proceedings of the 14th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2011), The George Washington University, Washington, DC, USA, October 5-7, 2011 by Kamran, S. and Haas, O.C.L, (2011)

1.5. Scope of the Report

The ITS@CU platform is a large system and involves various components and technologies. This report focuses mainly on the Semantic Agent-based Controls implementation aspects of the ITS@CU platform.

Not all aspects of the development details of ITS@CU are covered in this report due to the report length constraint. However, the appendices section includes the details of some important components/sub-components.

1.6. Report Organisation

The thesis is organised in different chapters as follows:

1. Research Introduction: Introduces this research in terms of its Aim & Objectives, Novelty and research methodology and contributions.

2. Background & Literature Review: This chapter outlines the research background, and reviews the theoretical and technological aspects of the areas related to this research such as such as ITS, Agent oriented technologies, Semantic Agent-based Controls and SOA. It also presents a review of related work, systems and studies related to this research.

3. Road Network Model Development: This chapter presents the Road Network Model which is used as a foundation for the design of the Agents and ITS@CU platform components.

4. Agent-based Controls Design & Organisation Structure: This chapter describes the design and modelling of the Agents in the ITS@CU platform. It covers the detailed design of the three main Agent types, their sub-types/roles, capabilities and their Organisational structure.

5. Agent Communication structure: This chapter describes the structure of the Agent Communication Layer used by agents to communicate and coordinate using the ontology based message instructions.

6. Agent Semantics, Ontologies and Rules structure: This chapter describes the design of the Ontologies and Rules used in multi-Agent communication layer to describe the semantic behaviour of the Agents and facilitate in decision making.

7. Implementation of ITS@CU platform: This chapter presents the technical design and development description of the ITS@CU platform relevant to the implementation of the Agent design, communication and semantic behaviour approaches described in chapters 4, 5 and 6.

8. Evaluation & Analysis: This chapter evaluates the overall research approach used in the implementation of the platform. It presents an analysis of the ITS@CU platform by simulating different traffic scenarios and test cases.

9. Conclusion & Further Work: This chapter concludes the thesis by discussing the outcome of the overall research and its objectives. It also outlines future implementations and suggestions.

Appendices: This section presents additional materials and outlines supporting literature. It also contains the various published papers; and sources code and ontologies in an encrypted CD.

See *figure 1.2* illustrating the flow of the thesis and report organisation.

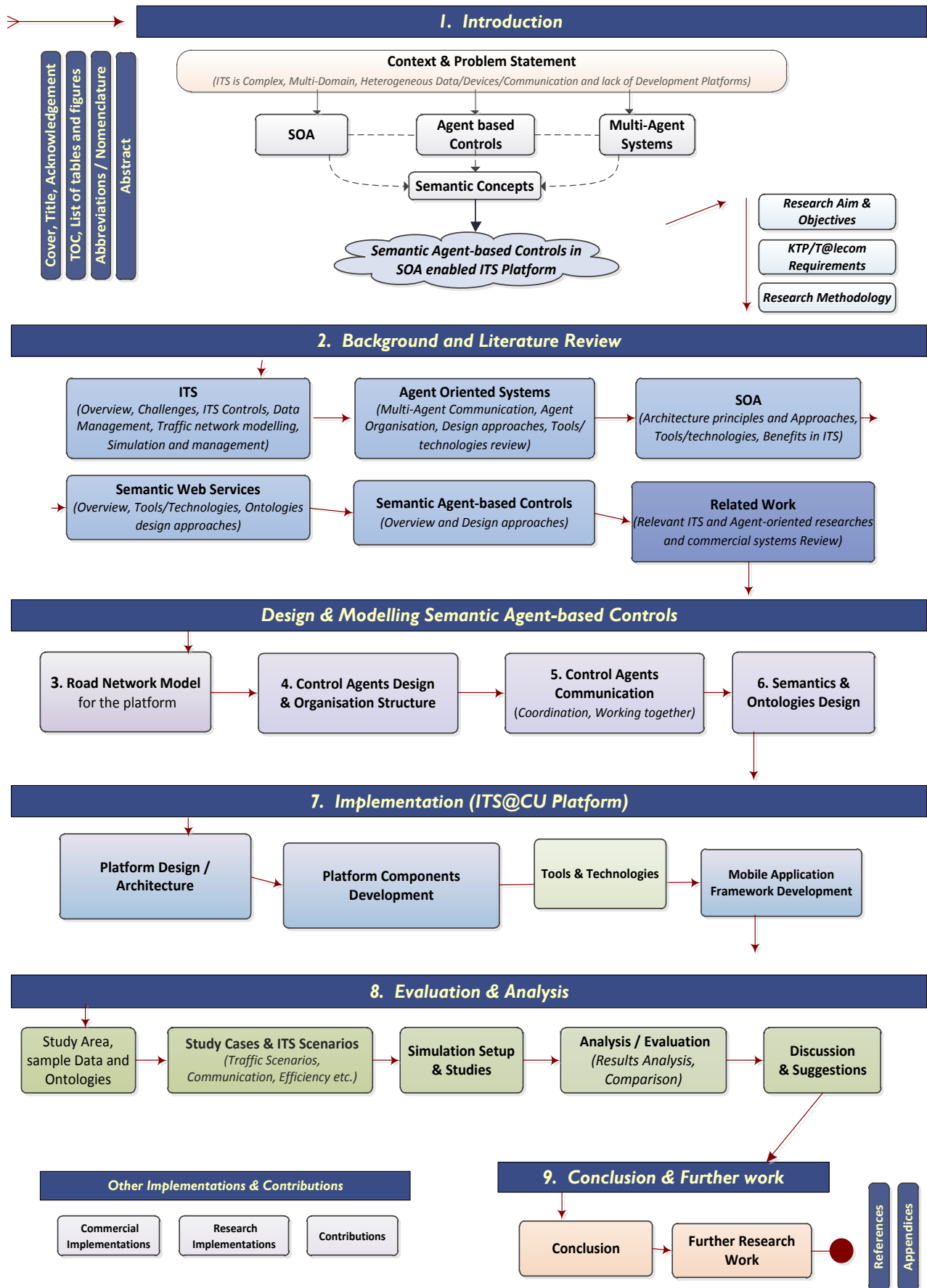


Figure 1.2: Thesis Flow and Report Organisation

Chapter 2

Background & Literature Review

The previous chapter introduced this research in terms of its aim & objectives, novelty and research methodology. The academic contributions made to the relevant research area (papers published and research projects engagement) and the relevant industry contributions (commercial implementations of the research outcome) were also stated.

This chapter outlines the research background, and reviews the theoretical and technological aspects of the areas related to this research such as:

- Intelligent Transportation Systems (ITS) and telematics
- Agent oriented systems and technologies
- Semantic Agent-based Controls
- Service Oriented Architecture (SOA) and grid computing
- Communication methodologies
- Relevant tools/technologies

The later part of the chapter presents the review of related work, researches and commercial systems.

2.1. Intelligent Transportation Systems (ITS)

2.1.1. ITS overview

The increasing level of traffic volume and limited growth in the capacity of the traffic infrastructure has resulted in various social, economic and environmental problems. One of the major problems is congestion especially the non-recurrent type of congestion, which is difficult to manage and plan for the authorities, and causes unexpected delays for the commuters and the deliveries of goods and services (Ozbay, 1999). It is also the main cause of road traffic incidents. The ever increasing consumer demand for the faster delivery of goods, services and shorter travel times mean that congestion and associated traffic management problems have become a noticeable obstacle to the economy and workers' productivity (Tang and Wang, 2006).

This trend has put the already deteriorating traffic infrastructures under immense stress and the efforts by the authorities to improve the infrastructure are unable to keep up with the pace of the growing transportation demand.

In order to address these problems, a variety of measures have been taken over the years such as infrastructure expansion, public transportation improvements, road sharing programs, variable speed controls, counter flow, and technological enhancement such as active traffic management, adaptive traffic lights signalling and broadcasting traffic information (Bazzan, 2005).

However, the volume of traffic is constantly increasing worldwide and it is likely to continue in the future which means that the traffic problems will become even worse compared to the current state if drastic measures are not taken on a continuous basis. The problems and challenges faced are indeed complex, intertwined and require coordination from all parties involved in the operation, management and research of the transportation system to achieve efficient, integrated, safe and environmentally responsible transportation.

This research explores the area of Intelligent Transportation Systems (ITS) which offers a promising future to revolutionise the modern transportation infrastructure's implementation and planning and to resolve traffic problems including providing a reduction in accidents and traffic congestion.

ITS refer to a wide range of systems as well as research efforts intended to enhance the productivity and safety of transportation systems by using advanced information and communications technologies. ITS is a wide ranging area and includes various technologies and applications such as advanced information processing (computers), communication technologies (wired-wireless), electronics devices (sensing and control), electrical equipment and strategies for management and planning traffic infrastructure – *all in an integrated manner* - to achieve the following transportation benefits.

- Improved Safety
- Reduced Congestion and better mobility
- Improved energy efficiency
- Enhanced economic productivity

- Reduced environmental impacts
- Reduced infrastructure costs
- Reduced operational cost by efficient automation
- Improved monitoring & management of traffic flows, incidents and efficient response
- Improved data collection on traffic flows, goods carried, carriers, drivers

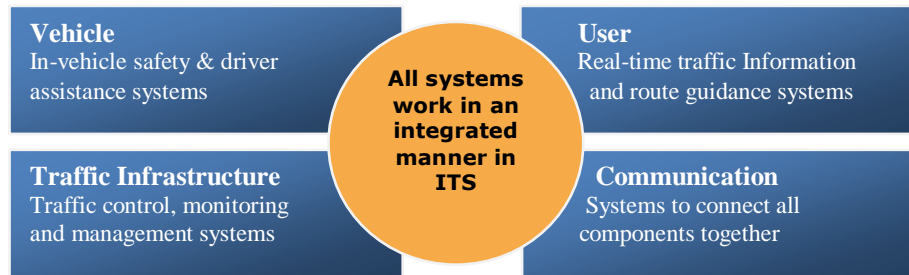


Figure 2.1: ITS Components

Vehicle: It is one of the main ITS components, which focuses on improving “in-vehicle intelligent systems” such as collision detection, avoidance and warning systems, advance driver assistance and navigation system, tracking, lane detection, safety monitoring, speed control, vision enhancement etc.

User: Focuses on enhancing the system providing assistance and real-time information to the users of any ITS related system such as navigational assistance systems, traveller information and route guidance according to the current traffic conditions, driver performance monitoring, and drivers’ comfort and safety improvement system.

Traffic Infrastructure: Intelligent traffic infrastructure is vital for the success of any ITS implementation as it provides a platform allowing efficient traffic monitoring & control, incident detection & response and other administration functions. It includes applications for:

- Monitoring weather & environmental conditions, collecting traffic data (rate of flow of vehicles, congestion)
- Detecting incidents, vehicle locations etc.
- Responding to incidents
- Management of planned and unexpected traffic events
- Controlling traffic signals and other such devices/sensors
- Administration of regulatory enforcement

Communication: Communication systems make the ITS work, as they link and provide the ability to exchange information between all the components. It allows data gathering for processing into intelligence, which can be then used efficiently to determine and activate appropriate response actions in the event of traffic incidents or faults.

During the last 5 years, ITS-based traffic infrastructures have been significantly developed due to the emergence of high-tech computing technologies and the integration capabilities of the new generation of traffic systems and devices. Most of the developed countries now have departments and organisations on both regional and national level, which are planning and implementing next generation of ITS-based architectures, standards and sub-systems.

Further review of ITS is included in “*Appendix C, section 1 and 2*”

2.1.2. ITS controls and traffic data management

The real-time data collection about the vehicles and road condition plays a vital part in managing, planning and forecasting the traffic flow. The data collected from various sources goes through various steps, systems and domains to provide the user level information. Over the years, the data collection methods have been changed considerably and the modern traffic infrastructures involve arrays of controls, devices and technologies to collect the traffic information.

The traffic data collections can be divided into the following two categories:

Road-based: This category involves the use of fixed devices and sensors on the traffic infrastructure, based on various methods and technologies each with their advantages and disadvantages in terms of reliability, cost, performance and data accuracy.

Device/Source	Description
Inductive Loops	These are metal detection devices installed along the roads to detect traffic flow situated mainly at the traffic lights. It is a reliable technology; however, the shift is now more towards cheaper alternatives such as infrared sensors (Turner et al., 1998). The accuracy of the data collected and exploited depends on the placement of the inductive loop which is not always located sufficiently close to the traffic controls.
Video recognition/CCTV	Arrays of cameras alongside roads mainly on motorways to monitor traffic condition. They are monitored by operators; however with advanced image processing techniques it is moving towards being fully automated. In Coventry a number of road corridors have been equipped with CCTV cameras for traffic monitoring and enforcement.
Active and Passive Infrared sensors	An alternative technology to inductive loops usually placed at traffic lights based intersections to detect vehicles. It uses thermal radiation for passive infrared and reflective signals for active infrared (Ozbay, 1999).
Weather sensors	Array of sensors to monitor weather conditions to assist traffic operators in managing the dynamic traffic flow.
Pipes	Two rubber tubes filled with liquid are placed across the road separated by a short distance. Sensors placed on the end of pipes detect the change of pressure when run over. They can determine the speed and type of a vehicle.

Laser	Variety of laser based technologies to detect vehicle information
Pulse-Doppler radar	Active detection technology to detect objects including their bearing, range, and altitude and also measuring their radial velocity (range-rate).
Ultrasonic sensors	Transmits low pressure waves of sound energy below human audible range to provide vehicle count, presence and lane occupancy information. (Klein, 2002)
Acoustic sensors	Array of acoustic sensors to detect acoustic signals by vehicles mainly used for vehicle count and multi-lane road monitoring
Electronic Toll Collection Sensors	Roadside beacons scanning signals from tags on the windscreen. Most often it is used for congestion charge systems but can indicate the number of cars which have entered a particular area.

Table 2.1: Traffic Data Collection Sources

Vehicle-based: This category involves the use of various in-vehicle detection technologies. Mainly it is used in vehicles by traffic authorities, police and in some cases public transport vehicles as probe vehicles. Probe vehicles with dedicated equipment provide real-time traffic data of route journey times and such data stored over a period of time provides a good source for analysing, planning and predicting journey times on different routes (McDonald and Li, 2006).

The use of navigation and tracking systems is also becoming widespread. Almost all the major car manufacturers are installing GPS receivers as a standard item; alternatively it is available as relatively-cheap add-on. It is likely that within few years' time almost all vehicles will be equipped with GPS receivers and navigation systems in most of the developed countries. Beside the purpose of navigation, GPS technology is also widely used for vehicle tracking systems where the vehicles are equipped with General Packet Radio System (GPRS) or Third Generation (3G) based vehicle modems or mobile devices to transmit real-time GPS data. This has opened a new dimension in the area of traffic data collection where vehicles themselves provide up to date location, speed and other such information to a central system; which in combination with cumulative traffic information can provide much better traffic flow management.

Data fusion technologies

ITS based traffic infrastructures involve heterogeneous data from a wide range of sources such as sensors, devices/equipment, control systems, GIS, manual and other application sources, and in various formats (Wu et al. 2005). The data collected requires highly efficient fusion techniques in order to provide comprehensive and accurate information required for further processing. Several data fusion algorithms and techniques have been developed and applied, individually and in combination, providing users with various levels of informational detail (Sharma, 1999). Selecting a suitable fusion technique depends on the type, level and amount of information involved.

The traffic management applications involving pattern recognition, artificial intelligence, neural networks, fuzzy logic, Figure Of Merit (FOM), expert systems and Kalman Filtering are already becoming common. Other algorithms, such as Bayesian inference, Dempster-Shafer inference, and voting logic, have also been used to detect and classify multiple source data for ITS purposes such as incident detection.

There are three basic categories or levels of data fusion (Linn & Hall, 1991) differentiated according to the amount of information they provide.

- Level one: Fusion of multi-sensor data determining the position, velocity, and identity of a target.
- Level two: High level of inference and delivering an additional interpretive meaning to the raw data.
- Level three: The highest level data fusion designed to make assessments and provide recommendations at user level.

The following table shows the different data fusion algorithms/techniques and methods at different levels.

Fusion Level	General Method	Specific Technique
Level one	Data association	Figure Of Merit (FOM)
	Positional estimation	Kalman filters
Level two	Identity fusion	Bayesian decision theory, Bayesian Inference, Dempster-Schafer Evidential Reasoning (DSER), Adaptive neural networks
	Pattern recognition	Cluster methods
Level three	Artificial intelligence	Expert systems, Blackboard architecture, Fuzzy logic

Table 2.2: Common data fusion techniques

Major ITS based projects using different data fusion techniques are discussed in detail in “*Appendix C, section 3*”.

In this research the data collection devices and controls were analysed to model their behaviour in developing agent-based controls and also for simulation in PDA based mobile applications mentioned in the implementation *chapter 7, section 7.4*. The data fusion techniques were investigated to develop an efficient data fusion mechanism for the ITS@CU platform which involves data from multiple controls. The author primarily focused on utilising level three expert System based fusion techniques. This was mainly achieved using SQL Server 2008 built-in data transformation services which are part of SQL Server Integration Services (SSIS).

2.1.3. Traffic light controls

Traffic signals at intersections or crossings are the most common and effective means of controlling the traffic flow. Poorly designed traffic signals can result in unnecessary and excessive delays. On the other hand, if appropriately designed, it can provide orderly movement and could actually increase the capacity of the intersection (Chawdry and Sadek, 2003). Traffic controllers can be classified according to the method in which they allocate green time for each phase and can be roughly classified into the following types of control:

Fixed-time control: A signal timing plan is selected according to a fixed schedule (e.g., time-of-day, day-of-week) from a set of predetermined plans, which were developed off-line on the basis of historical traffic data. The duration and order of all green phases remain fixed and are not adapted to fluctuations in traffic demand. The main drawback of fixed-time control is that it is not able to adapt itself as it is based on historical rather than on real-time data (Katwijk, 2008). Historical data is often not representative for the current situation such as:

- Traffic arrives at the intersection randomly, which makes it impossible to predict the traffic demand accurately
- Demand changes on the long term leading to “ageing” of the optimised settings
- Events, accidents, and other disturbances may disrupt traffic conditions in a non-predictable way
- Demand may change due to drivers’ responses to the new optimised signal settings (Diakaki & Kotsialos, 2003)

Actuated control: Signal timings (green/red phases) are extended or terminated depending on the current traffic demand usually using traffic detectors such as inductive loops to indicate the presence or absence of vehicles.

Adaptive control: A traffic control system that continuously optimises the signal plan according to the actual traffic load factors such as time, weather, and unpredictable situations such as accidents, special events, or construction activities. Adaptive traffic control systems continuously sense and monitor traffic conditions and adjust the timing of traffic signals accordingly. Adaptive systems, such as SCOOT and SCATS, have been around for a long time and have proven their worth in various places around the world. Using real-time traffic information, an adaptive system can continuously update signal timings to fit the current traffic demand.

The continuing growth in computational power enables control systems to further cater to the dynamics of the traffic system. Adaptive traffic systems are currently the most advanced and complex systems available. Whereas the working of fixed-time and traffic-actuated control systems is generally well-understood and is more-or-less standardised, this is not yet the case for traffic-adaptive systems. As traffic-adaptive systems operate on the forefront of what computers, monitoring equipment, traffic prediction, and optimisation are capable of, these systems significantly differ in their approach of traffic-adaptive control (Katwijk, 2008)

In this research, the study of signal controls was important for the simulation of Agent-based Controls representing traffic signal controllers. The behaviour of different types of traffic signal controls has been implemented for developing Agent controls and mobile simulation applications (described in *chapter 7, section 7.3.2.4*).

2.1.4. Traffic flow simulation

Traffic flow simulation systems and techniques are very important for analysing and optimising traffic flows and capacity. It is helpful in planning, implementing, and estimating traffic and its environmental effects. Simulations can be performed on a small part of the roads to a large traffic network/area. Simulation becomes more complicated with the increase in the size of the traffic network (small area, town, big part of the city etc.) and the level of detail.

Simulation systems and algorithms depend on traffic flow controls and models. There are three basic elements typically used for describing traffic stream: flow, speed and density (Chowdary and Sadek, 2003).

Elements	Description	Parameters
Flow (q)	The rate of flow, measured in the number of vehicles per unit time $q(x,t)$	Maximum capacity qm
Speed(u)	Distance vehicle travelled during a time	Free Flow Speed u_f Optimum Speed u_o
Density (p)	The number of vehicles per unit length of road $p(x,t)$	Jam Density p_j Optimum Density p_o
Velocity (v)	Mean velocity of the traffic flow $v(x,t)$	
Headway	Time or distance gap between two vehicles in the traffic stream	Space Headway d Time Headway b

Table 2.3: Flow elements

The traffic flow elements/parameters are related to each other by following equation

- Flow = speed * density ($q = u * p$)
*i.e. 10 vehicles in 1 mile stretch (density $p = 10$) and 50 mph average speed of vehicles ($u = 50$)
means the flow $q = 500$ vehicles per hour*

The traffic flow elements or parameters play an important part in the traffic modelling. The following are classifications used for describing traffic models:

- Scale of values (continuous, discrete, semi-discrete)
- Level of detail (microscopic, mesoscopic, macroscopic)
- Predictability of an algorithm (deterministic, stochastic)
- Source of data (analytical, simulation)
- Processing area (networks, stretches, links, intersections)

There are two general approaches for modelling a traffic flow:

Macroscopic models focus and analyse the traffic flow–density relationship for a traffic stream and its behaviour.

Microscopic models focus on describing and analysing the behaviour of individual vehicles and driver behaviour. Microscopic models represent an individual's behaviour, and have the flexibility to be adapted to the changes of road environment.

Traffic simulation systems are based on different data flow models and can be divided into the same categories microscopic, mesoscopic and macroscopic levels based on either continuous or discrete time approach. The macroscopic simulators involve analysing traffic flow where the microscopic approach focuses on analysing the behaviour of the individual cars/drivers. Mesoscopic mostly refers to approaches involving elements of both macro and micro or intermediate between both levels of simulation.

Researchers and the traffic system providers have always pushed the simulation towards more and more microscopic level, where the individual entities in the traffic network from vehicles to road based sensors could be taken into account during the simulation. Various technologies have been used to achieve this goal such as objects or agents that can be programmed to interact in a very natural way to produce accurate models of traffic flow behaviour (Kosonen, 1996). One interesting research area to complement microscopic simulation is Multi-agent systems. It allows the development of detailed microscopic simulations where each vehicle is controlled by an individual agent.

There are different types of models employed by traffic simulators based on traffic control strategies, area of coverage and level. Some of the well-known traffic simulators include VISUM, VISSIM, SOUND, MITSIM, SATURN, NETSIM, BOX Model and CORISM. A further review of these simulation systems is included in *“Appendix C, section 5”*

In this research, a Multi-Agent approach has been used for ITS@CU to simulate the behaviour of semantic based agents, traffic controls and vehicles. The benefits of using a Multi-Agent approach have been presented in studies by (Ehlert, 2005; Bazzan, 2005).

2.1.5. Traffic network modelling and mapping

A typical traffic infrastructure consists of various elements such as the network of roads/intersections, points of interest (petrol stations, services, parking etc.), on-road or road-side objects and equipment (traffic lights, signs, surveillance devices/sensors, information display etc.), services locations (e.g. tolls) and control centres. All these elements are distributed geographically and located on an identifiable address point on a map. Therefore, ITS-based systems dealing with the traffic network infrastructure requires an efficient mapping and modelling strategy in order to simplify the infrastructure planning and design, and improve the communication between the traffic elements.

Digital mapping technologies are widely used for structuring the traffic infrastructure's geo information in terms of

their location, distance and other properties. Generally the attributes of map elements and the road network model can be categorised as follows (Clemens Portele, 2007):

- Navigation: Arterial classification, dividers, barriers, one-ways, speed limits, road signs, turn restrictions, ramp signs, time of day and flow restrictions
- Geometry: Links, nodes, shape points, relative elevation, connectivity
- Path: Street names, route number and address ranges
- Points of Interest: Hotels, restaurants, tourist attractions, parking, transportation terminals etc.
- Cartography: Railroads, rivers, canals, lakes, golf courses, shopping centres, woodlands etc.
- Administrative: Country, state, county, city, and post codes etc.
- Traffic Codes: Database elements that provide map displays of traffic problems and dynamic route guidance when used in conjunction with real time traffic incident information

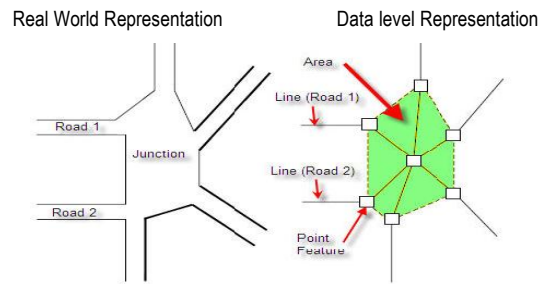


Figure 2.2: Real world junction and its map representation as simple features

There are various digital map formats developed by different providers such as NAVTEQ, ESRI, Garmin, Google, MapQuest, Genasys, GeoMicro, GeoSpatial Technologies, eSpatial, Maporama, MapInfo, MapSolute, Microsoft MapPoint, Mobiliris, Telogis, Autodesk deCarta and Telmap. Most of the maps are in proprietary formats and not readable without their special parser.

In this research, a road network model was developed to support the ITS@CU platform. The model was based on SCOOT (Siemens Mobility, 2009), and adapted for the proposed agent based controls and grid approach (discussed in *chapter 3*). Mapping technologies were important for the ITS@CU platform allowing location aware multi-agent communication based on specific area or location between the Agent-based Controls. Therefore, different mapping technologies and formats/standards such Geographic Data File (GDF), Geography Markup Language (GML), GPS and GIS were explored as part of this research. GML provides a good format and it is used as a basis for map data representation; however it was customised for the ITS@CU platform. For simulation and monitoring, Microsoft Bing was used due to its extensive map support, .NET compatibility and support for map services such as customised tiles, map layers, geo-coding, and location, traffic and direction information.

A further review of the mapping technologies relevant to this research is included in “*Appendix C, section 6*”

2.1.6. Incident management

In any ITS system, efficient management of traffic incidents is vital for keeping the traffic flow normal and to avoid unnecessary congestion by quicker incident detection and response. The process of incident management consists of four sequential steps (Ozbay and Kachroo, 1999):

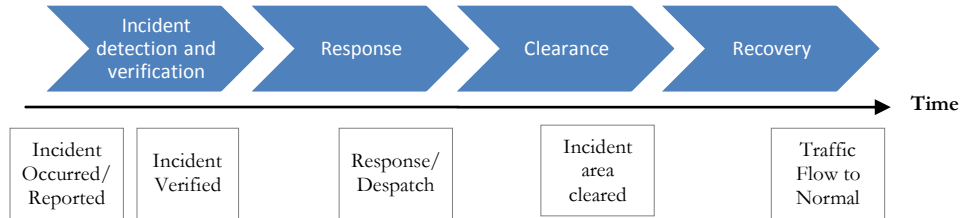


Figure 2.3: Incident Management Process

Incident management is a complex process and involves the coordination of activities from traffic authorities to restore the traffic flow to normal after the incident. From a technological perspective, the main problem lies in the efficient data collection, surveillance systems accuracy and malfunctions. It is therefore important that the data obtained is validated for its accuracy.

Incident detection and verification: Timely and accurate detection of a traffic incident and the verification of its existence in terms of time, location and nature is the most important step in the process of incident management. Generally, most of the incident detection systems follow two steps in order to determine the occurrence of an incident:

1. Determine the existence of congestion: Using the traffic data from on-road surveillance and monitoring systems e.g. CCTV, inductive loops, probe vehicles, sensors as well as reports by drivers
2. Analysing the data to verify the incidents: using incident detection systems and algorithms e.g. Automated Incident Detection (AID).

There are various Automated Incident Detection (AID) algorithms and systems, using different surveillance technologies and different methods to analyse the traffic data. AID algorithms are generally Model-based, Image-based, Prediction-based, and pattern recognition based. AID systems and Algorithms are evaluated using quantitative measures such as Detection Rate (DR), False Alarm rate (FAR) and average Time to Detect (TTD).

Incident response, clearance and recovery: Incident response comprises of the following time based components and reducing the time in any of the components clearly reduces the overall response time.

- Detection/Verification Time (T_1)
- Dispatch Time (T_2)
- Travel Time from dispatch location to incident location (T_3)
- Incident Clearance Time(T_4)

$$\text{Overall Response Time } (T_R) = T_1 + T_2 + T_3 + T_4$$

The incident response process consists of the following steps:

- Incident characterisation/classification: categorise incident based on factors such as type, location, fatalities/injuries, number of vehicles, number of lanes, weather condition, infrastructure damage, fire, hazardous materials, and cargo spill.
- Service Identification: Devising complete package of response services/actions by traffic operators based on incident characterisation
- Notifying relevant authorities: Ambulances, Fire, Police, Highway/road maintenance team etc. as per requirement
- Clearance/Traffic flow normalisation: Actions to clear the incident scene (debris/vehicle removal, passenger/injured evacuation, evidence collection, informing authorities and traffic restoration actions)

Often the authorities decide to divert the traffic flow in order to alleviate congestion caused by major accidents. The diversion is a tricky process for traffic authorities due to practical as well as political constraints (avoid residential area) and the impact it has on other roads and could result in congestion on the other road.

In this research, various AID approaches (review presented in “*Appendix C, section 7*”) were investigated from a technical point of view to design Agent-based Controls with the ability to respond to emergent traffic situations and make intelligent decisions to restore the traffic flow. Most particularly, the author studied the approach used in SCOOT (by UTM Control Room, Coventry city council) for dealing with traffic issues and planning. Another important aspect considered was the dynamic signal adaptation method for traffic flow optimisation using Agent-based Controls for analysing the traffic conditions. As part of the research, various strategies for traffic diversion and alternative route generation techniques were analysed, however, the implementation of route generation functionality in the ITS@CU platform was achieved using the Microsoft Bing API web service which provides customised route generation as a service.

2.1.7. ITS in the context of this research

The field of ITS is constantly evolving and researchers and system providers are exploiting different kinds of technologies and concepts from different principles to innovate the transportation infrastructure and vehicles. Some of the major areas of interest in computer science include AI, neural networks, distributed computing, multi-agent systems, grid computing, computer vision, pattern recognition, machine learning, data mining and intelligent controls. Other areas include applied cognition, HCI/user interface design, and even psychology for monitoring the driving behaviours, and predicting the drivers’ physical and mental states.

Currently, various ITS-based systems and controls are implemented throughout the developed world, which are significantly improving the transportation safety, mobility, and productivity (Tang et al. 2006). However, the full potential is still far from implemented and we have a long way to go. As more new concepts, methods, tools and technologies continue to emerge in the field of communication, information, automation and electronics, the more

ITS will progress towards achieving its vision of revolutionising the traffic system and dramatically reducing its associated problems.

This research investigates various methods and technologies from the perspective of a new generation of vehicles and roadside traffic controls which can interact autonomously and intelligently without the need of significant infrastructural changes. The following are some of the ITS promises and visions for the future of transportation relevant to this research:

Smart vehicles on Smart Roads: Modern vehicles are getting smarter and smarter and they already include many different kinds of sensors, CPUs, software systems, and communication capacities (Li et al. 2005). In the near future, vehicle-2-vehicle and vehicle-2-infrastructure sensing and communication will become standard features. It will be used for assisting drivers/passengers in a more intelligent manner for improving the driving safety, efficiency and comfort. On the other hand, the transportation infrastructure will also become intelligent to efficiently manage the vehicle flows and improve safety.

Artificial Transportation Societies: It is an exciting new concept aimed to study the social effects of complex transportation problems. These societies are mainly based on multi-agent systems integrated with transportation models including analytical descriptions of traffic flow models as well as rule-based human and vehicle behaviours and social and natural events. It allows transportation activities to “grow” in a bottom-up fashion, providing an alternative to real systems for experimental investigations (Wang and Tang, 2004). This will provide a better way to test and evaluate different traffic management systems, and simulate system behaviours on a larger scale by using the artificial traffic infrastructures.

Intelligent travelling spaces: Intelligent spaces are environments which have the capabilities to monitor all internal events, and are able to communicate with their inhabitants and neighbouring environments. Such intelligent spaces would be the next natural step in the advancement of ITS infrastructure. Incorporating intelligent-space technology into transportation will help to lay the foundation for a connected transportation environment where vehicles and infrastructure components intelligently, and securely share information, self-manage efficiently and react to the occurring problems (Wu et al. 2005).

2.2. Agent Oriented Systems and Controls

2.2.1. Agent technology overview

An ‘Agent’ is an autonomous software entity that can interact with its environment i.e. other agents, hardware and applications. “Agents are computer systems with two important capabilities. First, they are at least to some extent capable of autonomous action – of deciding for themselves what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents – not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives: coordination, cooperation, negotiation and the like” (Wooldridge, 2009).

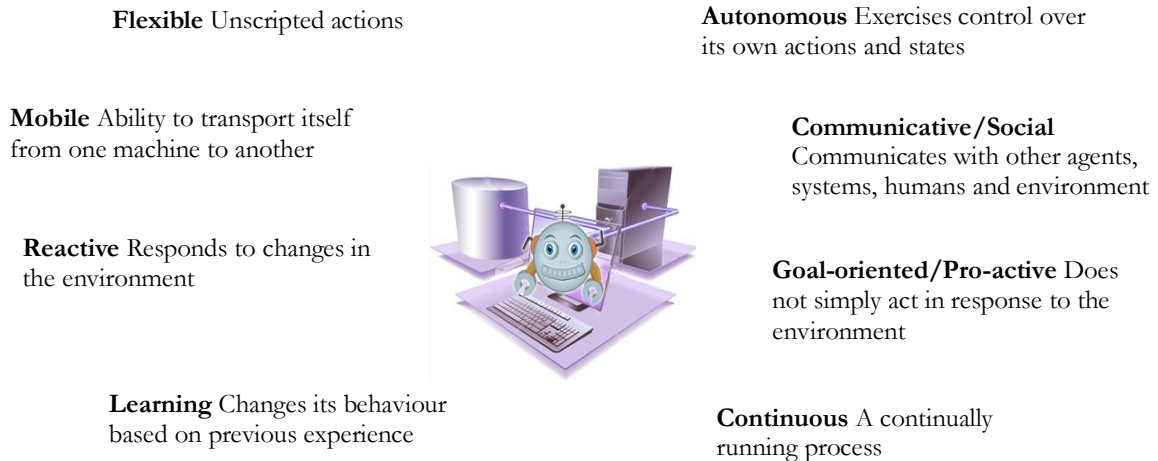


Figure 2.4: Agent properties

An agent must respond quickly to changes in the environment in order to be useful and act autonomously and where required communicate, negotiate and collaborate with other agents to achieve its goals. All Agent types satisfies the first four properties (reactive, autonomous, goal-oriented and temporally continuous) and the others are added as per application requirement (Buchanan, 2000).

There are weak and strong notions of software agents. The weak notion considers agents as a software-based computer model with Autonomy, Social abilities, Reactivity and Pro-activeness as their main properties (Oliveira, 1999; Wooldridge and Jennings, 1995). The strong notion of agents adds concepts like mentality, emotion and sociality to the definition of an agent in an attempt to simulate more human like behaviour (Castelfranchi, 1998). Agents are can be distinguished based on the level of their intelligence. Intelligence in an agent is usually defined by Autonomy, Intentionality, Reasoning and Learning.

Types of Agents

Reactive agents: These types of software agents are designed to react to the impulses from the environment that they work in. Although they can sometimes act as Intelligent Agents (reason or learn), their actions are triggered by incoming signals. They are comparatively simpler to design (Sanchez Passos & Rossetti, 2010). As compared to the deliberative/Intelligent agents which possess a reasoning model, i.e. they engage in planning and negotiation in order to achieve coordination with other agents, the reactive agents use request/response type actions (Simonin & Gechter, 2005).

Intelligent agents: These types of agents are close to artificial intelligence technologies, because they have an ability to adapt, learn and respond according to their environment. They are also referred to as BDI agents, as they act intelligently based on their beliefs (knowledge), desires (goals) and intentions (means). They are proactive and act on their own initiative to achieve their goals. They can reason over facts and scenarios and often learn in order to gain knowledge needed to fulfil their desires (Guerra-Hernández et al., 2004).

Fuzzy agents: These types of software agent are based on the fuzzy logic implementation i.e. it interacts with its environment through an adaptive rule-base. They are often considered as part of the Intelligent Agents.

Distributed agents: Such types of agents include the capabilities suited for a distributed environment i.e. they are designed to be loosely coupled and can be executed as independent threads and on distributed processors.

Mobile agents: Mobile Agents can relocate from one machine to another in order to perform their tasks more efficiently. It is a composition of software and data with an ability to move from one computer to another autonomously and continue its execution on the destination computer (Borselius, 2003). Mobile agents decide when and where to move, and they accomplish such moves through data duplication. During its movement, a mobile agent saves its current state, and transports the saved state to the destination host, then resumes its execution from that saved state. Web search engine crawlers are examples of mobile agents because they relocate from one cluster of the server to another (Milojicic, 1999). Mobile agents are ideal for occasionally connected environments (e.g. wireless) where they can operate in on off communication bases.

2.2.2. Multi-Agent System (MAS)

MAS refers to a system that consists of multiple agents, each with a certain function, that communicate and cooperate with each other in order to achieve the goal of the system as well as their own goals (Wooldridge 2009, Tweedale et al. 2007). MAS have their own problem solving capabilities and the agents are able to interact with each other in order to achieve an overall goal (Oliveira, 1999). Interaction may occur between agents and between agents and their host environment (Weiss, 1999). The nature of the interaction depends on the purpose of the system and actual situation but the point of the agent interaction is to distribute the complex tasks between specialised agents and therefore complete them more efficiently.

As agents are autonomous entities and tend to prioritise their own goals they have to be coordinated by the system to ensure its coherency (Nwana 1996). If not coordinated, agents can conflict, waste their efforts and eventually fail to achieve the system goal (Durfee 2004). The basic way to coordinate agents is by using joint commitments and conventions. According to the notion of joint commitment, if an agent agrees to perform a certain action, it can be safely assumed that he will perform it. Conventions ensure that agents in the MAS will act in the same manner (Jennings et al., 1998). MAS have the following characteristics (Lee et al., 2006; Shehory & Sturm, 2001):

- An agent has an incomplete information to solve a problem
- No global system control (may have central control only for arbitration)
- Data is decentralised
- Computations are asynchronous
- Agents that are autonomous and distributed
- Agents may be self-interested or cooperative

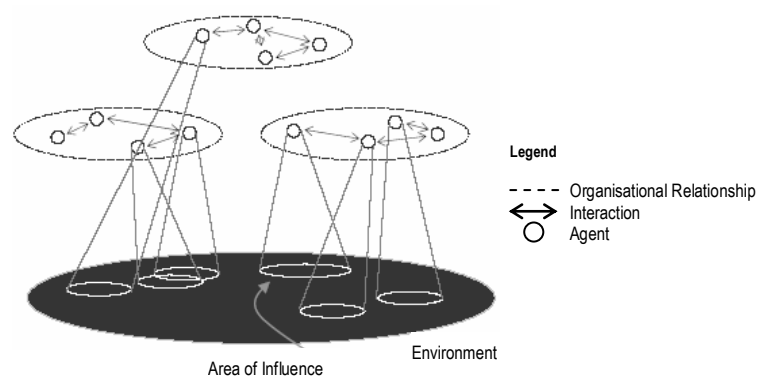


Figure 2.5: Typical structures of Multi-Agent Systems (Source: Adapted from Wooldridge, 2009)

In this research, the agent platform is based on the MAS approach due to the following rationale:

- A single agent that does everything could be constructed, but fat agents represent a bottleneck for speed, reliability, and maintainability. Dividing functionality among many agents provides modularity, flexibility, modifiability, and extensibility (Ozdag, 2007).
- A single agent may not carry or provide complete specialised knowledge which is spread over various sources (agents) and can be integrated for a more complete view when needed (Zhengping, 2006).
- MAS are ideal for distributed applications where agents can be designed as fine-grained autonomous components that act in parallel. Concurrent processing and distributed problem solving can provide solutions to many problems that up until now were usually handled in a more linear manner. Agent technology in this context can provide the ultimate in distributed systems (Andreas, 2005).

MAS require an appropriate environment to function:

- Multiple agents that are autonomous, adaptive, and coordinative
- An infrastructure specifying communication and interaction protocols
- Open with no (or limited) centralised control

In MAS, agents possess many functions such as programming, synchronising and prioritising tasks. They can collaborate or recruit resources, re-instantiate in different environments, store data, achieve messaging and communication etc. Most of the advanced agent based systems include hybrid of type of agents i.e. agents are designed to have the properties of multiple types for example distributed intelligent agent. In MAS environment, multiple type of agents can also interact and can achieve their own and cumulative goal. Agents can communicate over networks using multiple communication methods (e.g. HTTP, TCP), enter in new environments and adapt to the new environment.

2.2.3. Multi-Agent communication and coordination

The communication and coordination between agents is a key feature of Multi-Agent Systems (MAS). In a MAS environment agents must be able to communicate with their host/environment and with other agents to cooperate, collaborate and negotiate. In order to communicate, agents interact with each other by using special and commonly understood communication languages, called agent communication languages. In addition to a common language for agent communication, Agent-based controls require ontologies for understanding the communication semantics. Ontology is a representation of a set of concepts within a domain and the relationships between those concepts. When Agents communicate about a domain then it is necessary that there is an agreed set of common terminology that these agents understand for describing that domain.

The agent based languages provide both:

- Communicative acts (outer messaging layer/actions) *e.g.* KQML, ACL
- Content language (actual contents/ontologies/semantics within the messages) *e.g.* KIF

Knowledge Query & Manipulation Language (KQML): KQML was one of the earliest agent based communication languages purely based on speech-act-theory. “It is a language and protocol for exchanging information and knowledge that defines a number of performative verbs and allows message content to be represented in a first-order logic-like language similar” (Genesereth, 1994). KQML has actually provided the basis for FIPA ACL, a more advanced and widely used agent communication language (Labrou et al., 1999).

Knowledge Interchange Format (KIF): KIF is a language for the interchange of knowledge among disparate programs and KQML based agent-agent interaction. It has declarative semantics based on targeted domain in the form of expressions representation, with properties definition and their relationship for understanding between same domain agents. It is logically comprehensive i.e. contains the usual connectives of the first-order predicate calculus. It provides the representation of knowledge, reasoning rules and the definition of objects, functions, and relations within the communicative acts or messages in multi agent interaction.

Foundation for Intelligent Physical Agents (FIPA): “FIPA is an IEEE Computer Society standards organisation that promotes agent-based technology and the interoperability of its standards with other technologies” (fipa.org). It specifies the communication interfaces with different components in the environment with which an agent interacts (e.g. other agents, non-agent software, devices and services). The main emphasis is on standardising multi-agent communication using Agent Communication Language (ACL).

Agent Communication Language (ACL): ACL which is based on KQML, is an agent based communication language according to the FIPA standard, often referred to as FIPA ACL. A FIPA based agent message consists of a sender, receiver, performative and message contents. The performative is used to classify the message into meaningful low level intentions (fipa.org).

Further detail and a review of ACL relevant to this research is covered in “*Appendix D, section 1*”

According to FIPA, the content of an ACL agent message can be encoded in any content language. Ontologies encoded in agent messages can be defined in different formats/languages provided the involved agents can interpret and parse the message contents. More recently XML has become widely used for defining ontologies, which provides a highly flexible way of defining the semantics. Other ontology languages include Ontology Mark-up Language (OML), Web Ontology Language (OWL), Resource Description Framework (RDF) Schema, DAML+OIL and F-Logic. Further details are provided in *section 2.3*.

2.2.4. Security in agent-oriented environment

Security is vital for any system and network, however agent based systems require a higher level of security when compared to other distributed technologies (Borselius, 2003). It is mainly due to:

- Agent autonomy: Agent is autonomous and decides where it can go and what it will do, which deliberately or not deliberately can cause harm to the system in which it is executing or interacting with (Pitt et al., 2001).
- Information carrier: Agent can potentially compromise data by transferring its embedded data when migrating between hosts.
- Virus behaviour: An agent travels with its state of execution which means it can execute harmful operations.
- Outside interaction: Agents can execute commands as per instructions from external (non-trusted) sources.
- Migration independence: Travels across the network to any host independently without external influences (Buchanan et al., 2000).

In this research, the following security measures were considered for the multi-agent communication approach in the implementation of the ITS@CU platform:

- Multiple agents can co-exist and execute simultaneously however with some level of control and arbitration including agent-agent communication and with its hosts.
- Agent transport and acceptance mechanism from/to trusted hosts using identity verification process
- Ability to stop or limit agent despite the level of access or priority level of the agent.
- Ability to halt or freeze agents' execution states.
- Ability to constrain agents from interfering with each other in conflicting scenarios

2.2.5. Virtual Agent organisations

Agent computing enables a complex system (and its micro level components) to operate in a manner similar to a human organisation, thus allowing very flexible distributed computing and bringing the area of intelligent systems to another level. The major benefits of using agent computing especially in a complex system is the ability to self-organise, configure, manage, diagnose problems and even correct itself in different emerging situations/conditions, and thus to minimise the human interference and reliance (Madureira et al., 2008).

In MAS environments, agents form an organisation and have certain roles, responsibilities and set of tasks and objectives which they are assigned to perform and achieve (similar to human based organisation concepts). A Multi-Agent organisation has the following properties:

- Level of autonomy
- Power and authority
- Knowledge and skills
- Resources access

In order to achieve their tasks/goals, whether on individual level, team level or overall organisation level, agents:

- Communicate and coordinate with each other,
- Delegate tasks,
- Negotiate,
- Make decisions,
- Get commitments,
- Reach agreements

In an agent system, agencies (i.e. agent organisations) are main building blocks installed in each node of a networked system in which agents can reside and execute its operations. To facilitate the inter-operation of agents and agent systems across heterogeneous platforms, agencies designed to comply with agent standards are highly desired (Chen & Cheng, 2010).

In this research, these properties of MAS and virtual organisation formation approaches were important for the agent organisation in ITS@CU, which is a multi-agent platform with distributed Agents having different roles and organised in different agencies based on their domain level and area of influence (details are covered in *chapter 4*).

2.2.6. Agent oriented design approaches

The agent computing paradigm is rapidly emerging as one of the powerful technologies for developing large scale distributed systems to deal with the uncertainty in a dynamic environment (Chen & Cheng, 2010). Many researchers also argue that agent-oriented design represent the most important new paradigm for software development since object-oriented design (Luck, 2004). In recent years, agent oriented computing has further progressed and various architectures, models and methodologies have been developed for the design and implementation of agent based systems.

Agent architectures & design models

Agent architecture is defined as a particular methodology or framework for designing agent-oriented systems or individual agents. Agent design architectures are the fundamental engines underlying the autonomous components that support effective behaviour in real-world, dynamic and open environments (Luck , 2004; Bellifemine et al., 2007).

Agent architectures and design models can be divided into four groups:

Logic-based (Symbolic): Logic-based agents use traditional knowledge-based system techniques in which an environment is symbolically represented and manipulated using reasoning mechanisms (Bellifemine et al., 2007). The behaviour and desires of agents are based on logical deduction or theorem proving (Wooldridge, 2009; Poole et al., 1998). Logic-based reasoning comprises a set of knowledge representations (knowledge base) where logic is used to form axioms that describe facts (nuggets of knowledge) and their relationships (Lakemeyer and Nebel, 1994).

Logic-based agents are dependent on the knowledge base to perform their tasks. They are complex to design and comparatively slower than reactive agents due to deductive reasoning (i.e. processing of the knowledge base) (Russell & Norvig, 2010).

Stimulus–response (Reactive): Reactive architectures implement the agent decision-making capabilities based on a stimulus–response mechanism triggered by environmental factors (sensory data inputs). Unlike logic-based architectures, they do not have any central symbolic model and therefore do not utilise any complex symbolic reasoning (Bellifemine et al., 2007).

Reactive agents exhibit faster responses in dynamic environments and are often simpler to design than logic-

based agents (Sanchez Passos & Rossetti, 2010; Wooldridge, 2009). However, reactive agents lack the same level of reasoning as they do not employ models of their environment compared to other approaches. Also, it is a complicated process to design reactive agents to fulfil specific tasks in a complex environment where the situation changes all the time (Simonin & Gechter, 2005; Henderson-Sellers & Giorgini, 2005).

Belief-Desire-Intention (BDI): BDI architectures are probably the most popular agent architectures (Busetta, 2003). In BDI architectures, agents have certain mental attitudes i.e. beliefs, desires and intentions. ‘Beliefs’ refer to the information an agent has about the environment it is in, ‘Desires’ relate to the tasks that are allocated to the agent and ‘Intentions’ are desires that an agent has committed to achieve (Rao and George, 1995). In order to achieve its intentions, an agent has a plan library (stored plans) which represents the procedural knowledge.

Different agent-based systems have been implemented based on BDI such as Procedural Reasoning System (PRS) JAM (Huber, 1999), JACK (Howden et al., 2001), dMARS (d’Inverno et al., 2001) and JADEX, with a wide range of applications demonstrating the viability of the BDI model. One of the well-known BDI architectures is the PRS which is also relevant to the platform in this research. Its architecture is based on four key data structures: beliefs, desires, intentions and plans, and an interpreter (Ma et al., 2010). In PRS: Beliefs represent the agent’s information about its environment, which may be inaccurate; Desires represent the goals/objectives/tasks which are allocated to the agent; Intentions represent desires that the agent has committed to achieve; finally, plans specify the actions followed by an agent in order to achieve its intentions. These four attitudes (in form of data structures) are managed by the interpreter agent which is responsible for updating beliefs from environment observations, generating new desires (tasks) based on new beliefs, and selecting a subset of tasks from currently active desires to act as intentions. Finally, the interpreter selects an action to perform based on agent’s current intentions and knowledge (Bellifemine et al., 2007; Wooldridge, 2009).

BDI agents lack learning capability i.e. to learn from their past behaviour and adapt to new situations (Guerra-Hernández et al., 2004). BDI is composed of three attitudes (Belief-Desire-Intention) which are actually not sufficient for all types of agents systems (Rao and Georgeff, 1995; Busetta, 2003) hence it is not suitable for all types of agent system design.

Layered/Hybrid architectures: In layered architectures various layers are designed to deal with both the reactive and pro-active behavioural requirements of an agent. The philosophy of layered architectures is based on the assumption that agents need to produce both simple reactive behaviours (e.g. to avoid an obstacle in the case of robots) and pro-active behaviours (e.g. to plan for future actions). Agents that show more reactive behaviour perform relatively better in environments that are highly dynamic compared to pro-active agents that do better in

more static environments (Wooldridge, 1999). Layered/hybrid architectures allow both reactive and deliberative agent behaviour. To enable this flexibility, subsystems arranged as the layers of a hierarchy are utilised to accommodate both types of agent behaviour. There are two types of control flows within a layered architecture: horizontal (Ferguson, 1991) and vertical layering (Muller et al., 1995).

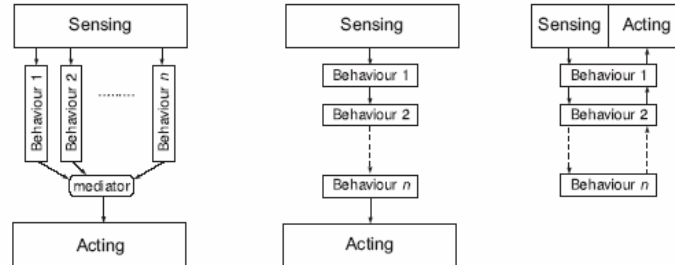


Figure 2.6: Layered architectures (Source: Bellifemine et al., 2007):
Left horizontal, middle vertical (one pass) and Right vertical (two pass)

In horizontal layer approach, the layers are connected to the sensory input and action output where each layer acts like an agent in itself. The main advantage of this is the simplicity of design i.e. single layer per agent's behaviour. However, since each layer is in effect an agent, their actions could be inconsistent prompting the need for a mediator function to control the actions. Another complexity and potential drawback is the large number of possible agent interactions/transactions between horizontal layers. Vertical layer architecture eliminates some of these issues as the sensory input and action output are each dealt with by at most one layer each (creating no inconsistent action suggestions). The vertical layered architecture can be divided into one pass and two pass architectures (as shown in figure 2.6). In one pass architectures, control flows from the initial layer that gets data from sensors to the final layer that generates action output. In two pass architectures, data flows up the sequence of layers and control then flows back down. The main advantage of vertical layered architecture is the interaction between layers is reduced significantly. The main disadvantage is that the architecture depends on all layers and is not fault tolerant, so if one layer fails, the entire system fails (Bellifemine et al., 2007).

In this research, combinations of Agent architecture and design models were adopted based on the specific role of the Agent and the requirements of that role. *For example*, a Reactive agents design approach was adopted for some types of Agents (Service Agents and resources provider agents) which do not require proactive properties to function i.e. these agents only react to incoming requests (Simonin & Gechter, 2005; Sanchez Passos & Rossetti, 2010). Control Agents were designed based on a logic-based approach as they can embed the controller's logic as part of the Agent functions. Most Operational Agents were designed to follow the BDI design approach which provides an ability to design agents with more sophisticated behaviour in a distributed environment (Ma et al., 2010; Vidal, 2010) making it an ideal choice for core operational agents in the platform (mentioned in *chapter 4,5 and 6*).

Overall, the architecture of the platform in this research was based on a hybrid architectural approach. A hybrid model allows the use of different underlying models/architectures for sub-components of the system; therefore, it was a better architectural choice for dealing with multiple architecture types of components/sub-systems in the ITS@CU platform. A hybrid approach of layering by combining both horizontal and vertical architectures provides a better approach for reducing the number of agent interactions (hence better performance and communication) and fault tolerance by eliminating the reliance on a single layer (Bellifemine et al., 2007).

Different types of agents were investigated for the design of Agent-based controls in the platform. Intelligent Agents are ideal for control applications (Vidal, 2010); however the proposed Agent-based controls also required the capability of agents to migrate between different domains and interact with other Agent-based controls to make decisions. Therefore mobile agent's capabilities were also useful in designing the agent structure and platform's capability to host/execute mobile agents. The concept of MAS was essential to this research as the ITS platform involved multiple agents working in synergy at different domain levels. The overall approach adopted in the research for Agent modelling was to design different types of specialised agents with the properties of mobile, reactive and intelligent agents or a combination depending on the Agent's role and purpose. The FIPA ACL based communication concepts of MAS were employed for multi agent cooperation and coordination; however the research takes the Agent design and MAS communication to another level by using semantic agent concepts based on a semantic web services approach (see *chapter 4, 5, and 6*). The Agents in ITS@CU are therefore flexible by design to promote agility and efficiency especially when it comes to domain specific problem solving in SOA based complex multi-domain environments.

Agent-based modelling and simulation

Agent-based modelling (ABM) and simulation uses multi-agent systems for the representation of social, economic, ecological and other similar systems in a software environment (Salamon, 2011). Agent-based modelling appears to be a promising technology that has the potential to improve the modelling techniques in complex systems (Dasheng et al., 2010). ITS systems are highly complex and distributed in nature which makes them suitable for Agent-based modelling (Andreas, 2005; Wang, 2008; Dasheng et al., 2010). Agent-based modelling and simulations are usually considered the same (Macal & north, 2007). Agent-based modelling can be used to design and represent the behaviour of traffic flow and infrastructure components where individual agents are programmed to control/simulate the behaviour of vehicles, traffic controls and other components.

Agent based modelling is different from an object-oriented modelling approach however object-oriented techniques usually provide a suitable foundation for the development of agent-oriented models (Bazghandi & Pouyan, 2011). Agent-based models are often confused with multi-agent systems (MAS) and it is important to

distinguish that ABM is primarily focused on modelling/simulating Agents, and MAS is the implementation of a system (with multiple agents).

ABM is not a general purpose modelling technique and requires the right level of description/detail to serve its purpose. ABM looks at a system at a component/unit level i.e. not at the aggregate level. Simulating the behaviour of all of the units can be time consuming and extremely computation intensive which means that the high computational requirements of ABM remain a problem when it comes to modelling large systems (Eric Bonabeau, 2002).

One of the interesting application areas of agent based modelling techniques is the simulation of traffic systems. As mentioned in *section 2.1.4*, Agent oriented technologies provides a very good approach for simulation especially microscopic level traffic simulation (Bazzan, 2005). Entities in the traffic network from vehicles to road based sensors could be taken into account during the simulation to produce accurate models of traffic flow behaviour (Kosonen, 1996). Multi-agent systems allow the development of detailed microscopic simulations where each vehicle is represented by an individual agent. There are different types of models employed by traffic simulators based on traffic control strategies, area of coverage and level. Some of the well-known traffic simulators include VISUM, VISSIM, SOUND, MITSIM, SATURN, NETSIM, BOX Model and CORISM.

A further review of these simulation systems is included in "*Appendix C, section 5*". The benefits of using a Multi-Agent approach in traffic simulation have been discussed in studies by (Ehlert, 2005; Bazzan, 2005).

Agent design methodologies

There are various methodologies for designing agent oriented systems. These methodologies typically consist of models and guidelines (Henderson-Sellers & Giorgini, 2005; Wooldridge, 2009). Some of the relevant methodologies reviewed in this research include:

Methodology	Description	Analysis aspects
Agent UML (Odell et al., 2001)	It is based on an object-oriented approach to designing Agent based systems by using Unified Modelling Language (UML). The notations in UML were extended to enable the modelling of Agent systems in order to support concurrent Agent interactions (multi-threaded) and the notion of 'role' to design different roles of agents in MAS.	<p>The familiarity of UML design models/notations and object-oriented concepts make it easier to design the agent based systems using Agent UML especially in systems where Agents are implemented using objects (Henderson-Sellers & Giorgini, 2005).</p> <p>It provides a good graphical representation of the agent system in form of UML diagrams.</p> <p>The main drawback is that it is greatly focused on object-oriented approach however the Agents implementation is not just limited to objects (Baurer et al, 2001).</p>
Gaia (Wooldridge, 2009)	It is a top down methodology and provides a systematic process from requirement specification to design phase in order to support the agent system implementation. It promotes moving from abstract to increasingly concrete concepts. The key Gaia concepts are roles, which have associated responsibilities, permissions, activities, and protocols. Roles can interact with one another using the protocols of the respective roles (Wooldridge, 2009)	<p>Gaia is tailored for agent design and analysis particularly to address the design of agent's flexibility, autonomous problem solving behaviour, multi agent interaction and organisational structure complexities (Wooldridge et al., 2000).</p> <p>The methodology is not easily scalable and provides weak support for open systems (Juan et al., 2002). Gaia requires detailed and somewhat complete requirements specification to design agent systems (Iglesias et al, 1999); but in most complex system the requirements are either only partially complete in the early phases or they change frequently (Henderson-Sellers & Giorgini, 2005).</p>

Desire (Framework for DESign and Specification of Interacting REasoning components)	It is a framework for the design and formal specification of compositional systems (Treur et al., 1995). It has been adapted for Multi agent systems and it views both Agents and its overall system as a compositional system (Brazier et al., 1997).	It is based on formal specification techniques which help in the agent design and modelling process by providing comprehensive specification. It has a graphical tool for assisting with the design process however it is out-dated. It also lacks support for BDI agent architecture.
Agents in Z	It is an Agent specification framework based on Z language (d’Inverno & Luck, 2001). It has a four-tiered hierarchical approach to designing agent based systems. The agents in this methodology are entities (objects) with attributes, capabilities and goals.	It is not very scalable and limited due to the Z language specification. It does not allow constructing reactive and proactive agents (Wooldridge, 2009).

Table 2.4: Agent design methodologies

A comprehensive survey on different agent methodologies has been provided in a study by (Iglesias et al, 1999) and an interesting analysis of agent methodologies is presented by (Henderson-Sellers & Giorgini, 2005).

2.2.7. Agent tools and technologies

Agent oriented computing is a relatively less established area, and the tools and technologies involved still lack standards. However, various standards are being discussed (Tweedale et al. 2007) and agent technology is getting attention from a few major software vendors such as Oracle/Sun and Progress Software. Currently, most of the commercial products/toolkits are proprietary and quite specific in nature.

A further review of relevant Agent Tools and Technologies is included in “*Appendix D, section 2*”

This research involved a wide range of development technologies in order to implement the different components of the ITS@CU platform. Various tools and technologies especially Java Agent Development Environment (JADE) and NetLogo were initially used for MAS experimentation and traffic simulation (see “*Appendix K*”).

JADE provides a good Agent development platform however it is limited to the Java platform and lacks SOA and integration with ITS based traffic controls/devices. One of the core research requirements was to focus on the Microsoft .NET development framework and C# due to its compatibility with T@lecom's other commercial systems. Therefore, the ITS@CU platform was developed from the ground up based on the Microsoft .NET development framework. It was obviously a big challenge and a vast amount of time was spent on the development of the platform however it was necessary due to the requirements. In addition to just the requirement, .NET technologies were also industry leading technologies and provided comprehensive functionalities and technical advantages over other technologies (reviewed in "*Appendix F and D, section 2*"). Some of the main benefits of .NET include a powerful IDE (Visual Studio 2008/10) and programming language (C#/VB.NET) support; native integration with SQL Server 2005/08 DBMS; an extensive range of APIs/libraries; and other technologies such Web Services/WCF, ASP.NET, native XML parsing and LINQ support.

2.2.8. Agent-oriented approach to ITS problems

ITS systems are highly distributed in nature with micro level components and multiple levels of decision making which makes the development, modelling and simulation very difficult. The domain of traffic and transportation systems is therefore well suited to an agent-based approach because of its geographically distributed nature (Andreas, 2005) and it's alternating busy-idle operating characteristics (Wang, 2008).

Agent technology's inherent distribution allows for a natural decomposition of the system/sub-systems into multiple agents that interact with each other to achieve a desired global goal (Parunak, 1999). Agent's properties such as collaboration, autonomy and decision making abilities provide a better way to implement automated traffic management and control systems. Agent-based transportation systems allow distributed systems/sub-systems collaborating with each other to perform traffic control and management based on real-time traffic conditions (Chen & Cheng, 2010).

The development (or improvements) of traffic infrastructures is costly and it must be carefully evaluated for its impact on the traffic before any actual work commences (Ozbay & Kachroo, 1999; Chowdhury & Sadek, 2003). Agent technology has an advantage to develop micro level simulation with individual agents simulating the internal behaviour, interaction and decision making capabilities of all the entities in traffic systems (Ehlert, 2005; Bazzan, 2005). A review of traffic simulation systems and techniques is also provided in *section 2.1.4*.

2.2.9. Difference between Agents and other technologies

Object-Oriented: The main difference between objects and agents is the autonomous nature of the Agent (Henderson-Sellers & Giorgini, 2005). An object is not autonomous as it relies on the host application and follows the class description defined and controlled by the application. Agents are not completely controlled by any application. Furthermore, agents are interactive entities that are capable of using rich forms of messages. These messages can support method invocation – as well as informing the agents of particular events, asking something of the agent, or receiving a response to an earlier query. Lastly, because agents are autonomous they can initiate interaction and respond to a message in any way they choose (Odell, 1999; Jennings et al., 1998).

Distributed/Concurrent systems: Similar to agent based computing, distributed/concurrent systems have multiple components which interact and run as separate entities or on separate threads, however there are two main differences (Wooldridge, 2002):

- Agents are autonomous entities i.e. capable of making independent decisions and not strictly hardwired in at design time.
- Agents are mostly self-interested entities, while the components/elements in distributed concurrent systems generally share a common goal of the system.

Artificial Intelligence (AI): Agent technology has been considered by many as part of AI, as it involves the use of various AI algorithms/methods especially when it comes to Intelligent Agents e.g. learning, planning, understand images, self-configure etc. However, Agent technology is much more than just the use of AI as it uses various non-AI technologies as well. It involves social aspects of systems interaction and communication, and has its own development methodologies and concepts.

An interesting analysis and discussion regarding the differences between agents and other technologies are covered in (Odell, 1999; Wooldridge 2009; d’Inverno & Luck, 2001).

2.2.10. Critical analysis and challenges of agents based approach

Despite the promising potential of Agent oriented approaches, there exist a number of concerns that should not be overlooked (Wooldridge, 2009). Agent technology still lacks development tools, protocols/standards, design techniques and methodologies as compared to more mature technologies such as object-oriented

(Henderson-Sellers & Giorgini, 2005; Luck et al., 2005). There are no widely-used software platforms for developing multi-agent systems that provides for example: a complete infrastructure required for agent message handling, tracing and monitoring, run-time management. Hence, a significant amount of time and resource is generally spent on implementing an infrastructure prior to the development of a multi-agent system (Wooldridge, 2009; Henderson-Sellers & Giorgini, 2005). This has been the case in this research and a new platform (ITS@CU) was specially designed as part of the implementation process (see *chapter 7* for the platform details and the reasons for implementing a new platform).

Agent researchers and developers often overestimate the potential of agent approach. It is important to understand its limitations and useful application areas/scenarios. Although agents have been used in a wide range of applications (Chen & Cheng, 2010), they are not a universal solution i.e. there are many scenarios in which other conventional software development approaches (e.g. object-oriented) can be more appropriate (Wooldridge, 2009). There advantages of an agent based approach should be clearly identified and understood prior to the actual implementation (Henderson-Sellers & Giorgini, 2005).

Security in agent systems is also a common concern due to the autonomous nature of agents which means they can behave like a virus and compromise data (mentioned in *section 2.1.4*). The dilemma is that if an agent platform is designed to be strict and more controlling then the agent becomes less autonomous which goes against the agent philosophy of having greater autonomy (Pitt et al., 2001). Therefore, the right balance of trust and security must be considered in open agent environments (Borselius, 2003).

Multi-Agent systems by design tend to be highly multi-threaded systems which presents its own complexities and challenges usually associated with concurrent and distributed systems (Vidal, 2010) i.e. concurrency and coordination issues of agent messages/process flows and distributive resource (host memory/CPU) allocation issues (d’Inverno & Luck, 2001; Chen & Cheng, 2010).

The design of individual agents is a challenging process and adopting an appropriate type and model required for agents’ roles is not a straight forward process (Macal & north, 2007). An incorrect balance of having too many or too few agents can also lead to an inefficient system (Vidal, 2010). Too many agents in a system can result in communication overheads, resource limitations, concurrency and coordination issues. By contrast, too few agents or fat agents (i.e. agents performing multiple roles) can negatively impact the system’s performance and its reliability i.e. if one agent fails then all the functions associated with the multiple roles of that agent fails (Ozdag, 2007). The design of an agent system without such considerations often leads to an implementation failure (Chen & Cheng, 2010; Tapia et al., 2008; Henderson-Sellers & Giorgini, 2005).

Readers interested in further information analysis on drawbacks and challenges of agent technology can refer to (Luck et al., 2005 and Wooldridge, 2009).

2.3. Ontologies for Agent communication

2.3.1. Ontology overview

In addition to a common language for agent communication, Agent-based controls require ontologies for understanding the communication semantics. Ontologies represent the domain knowledge as concepts and their relationships. When Agents communicate about a domain then it is necessary that there is an agreed set of common terminology that these agents understand for describing that domain.

Ontologies can be defined in any format or language as long as they are understood by the agents i.e. they have the capabilities to interpret the message contents. According to FIPA (fipa.org), the content of an ACL agent message can be encoded in any ontology definition language (e.g. KIF). More recently XML has become widely used for defining ontologies, which provides a highly flexible way of defining the semantics (Sabou, 2006). Other popular ontology languages include Ontology Mark-up Language (OML), Web Ontology Language (OWL), Resource Description Framework (RDF) Schema, DAML+OIL and F-Logic (Cardoso, 2007).

2.3.2. Ontologies design

Ontologies can be defined for a system at different levels classified as follows:

Foundational ontologies: Top-level conceptualisations that contain specifications of domain and problem independent concepts and relations based on formal principles derived from linguistics, philosophy, and mathematics.

Generic ontologies: Contain generic knowledge about a certain domain such as Telematics, medicine or biology. These domain concepts are often specified in terms of top-level concepts thus inheriting the general theories behind the top-level concepts.

Domain ontologies: Specific to a particular domain and have the lowest reusability outside of the domain.

Well defined ontologies are important for agent communication languages to reduce errors during the communication between the agents (Vidal, 2010; Luck et al., 2005). The more thorough the ontologies, the better the Agents are able to understand what a data resource is and how it relates to other data and resources. The ontology design for Semantic data representation in web services and Agent communication in ITS@CU is covered in *chapter 5 and 6*, and the implementation description is covered in *chapter 7*.

2.3.3. Ontology languages and technologies review

Ontologies are explicitly specified in a formal language such as Resource Description Framework (RDF), RDF Schema (RDFS), DAML+OIL, and Web Ontology Language (OWL). Currently, both OWL and RDFS are widely used however OWL is becoming more popular (Cardoso, 2007). Both languages have their strengths and which to use is entirely up to the designer/developer's own preference, experience and also the application area/type (Antoniou & Harmelen, 2009; Cardoso, 2007).

RDF only provides a model and syntax to describe resources; however it does not specify the semantics of the resources, so to define semantics RDFS and OWL are required (Altova, 2007). RDFS provides support for rich semantic definitions however it has a limited support for the specification of local usage constraints (i.e. structural, cardinality and data type constraints) (Hunter & Lagoze, 2001).

OWL is derived from DAML+OIL and builds on RDF concepts (w3c.org) and it is more expressive than RDFS with a larger vocabulary to describe classes and properties (Horrocks et al, 2003; Antoniou & Harmelen, 2009). It is based on Description Logic which provides the basic representation features of OWL with well-defined construct meanings. OWL also includes features which help it to integrate into the mainstream web technologies such as the Internationalized Resource Identifiers (IRIs) as names, XML Schema data types, and ontologies as web documents, which can then import other OWL ontologies over the Web. (Antoniou, 2009)

Ontology building processes still lack guidelines to facilitate what knowledge ontologies should contain and what design principles they should follow (Sabou, 2006). This is the main reason why ontologies described for web services/applications even in the same domain are vastly different which then hinders the reusability of the ontologies. Ontology building is a difficult and generally time consuming process with less automation support for large and changing textual data collections (Horrocks et al, 2003; Sabou, 2006).

A further review of RDF, RDFS and OWL is included in "*Appendix E*".

There are several ontology editors available such as Protégé, Swoop, OntoEdit and Altova SemanticWorks. However, Protégé is the most widely used editor and its users share is about 62.8% (Cardoso, 2007). Further detail is provided in "*Appendix F, Ontology Editors*".

In this research, ontology languages were explored in detail, especially OWL, however the ontology described for the agent communication is customised and defined in an XML format to suit the ITS@CU platform. XML was preferred due to its native support in SQL Server 2008 and .NET parsing capabilities. (Other benefits of using XML are mentioned in *chapter 5, section 5.2*). Primarily, Protégé software was used for designing ontologies in OWL, and then these ontologies were converted from OWL into XML for further customisation to adapt to the ITS@CU Agent communication structure.

2.4. Semantic Agent-based Controls & Services

2.4.1. Agent-based Controls

Any control system can be viewed as an agent (Wooldridge, 2002), for example a simple thermostat with sensor to detect low/high temperature (of the environment) can trigger the heating/cooling system to start/stop based on the current temperature. In more complex systems, multiple controls can be agents themselves or governed by different types of agents over the network. Therefore, the central concept behind agent-based controls is a type of control which is governed by an agent or agents (of any type).

In software terms, these agents are the control algorithms, or actual code logic or programs which are responsible for executing certain actions based on the environment. Agents based controls therefore can be:

- Purely code level agent entities over the network controlling single or multiple controls
- Local, static level sensors/equipment/systems
- or even a combination of both

In large distributed systems involving many control devices, if the local system controls the functionality of the devices not only does it require more memory and hardware (hence cost), but it also increases the complexity of the overall system. On the other hand, a network based agent can control multiple controls, and provide extra functionality and intelligence to the overall system and leaves the local controls to do basic computation on a local level. It allows decomposition of a control algorithm into many simple task-oriented control agents distributed over a wide area network. In this way, a network-enabled device can operate on a “control on demand” basis i.e. it will need to host only the operating agents, not all the possible agents required for its operation (Wang, 2005).

2.4.2. Semantic web services

Web service technology has its limitations and fails to achieve the goals of automation and interoperability because they require a well-defined service interface involving humans to write the description (McIlraith et al., 2001). WSDL specifies the functionality of the service only at a syntactic level allowing these descriptions to be automatically parsed and invoked by machines, however the interpretation of their meaning is still largely human (Martin et al., 2004). In order to overcome the limitations of traditional web services, the concepts from semantic web offers an ideal approach by augmenting the service descriptions with a semantic layer to facilitate automatic service discovery, composition and execution (Maleshkova, 2008).

Semantic web services implementation requires adding semantic metadata (data that describes data in the form of ontology) to the information resources, allowing computers to effectively process the data based on the

semantic information that describes it (Altova, 2007). Semantic web's current research focus is more directed towards the support of intelligent data exchange where the information that is being annotated is not unstructured text but rather semi-structured information available from databases or exchanged between web services (Sabou, 2006). The role of the semantic annotations is to support merging, integrating and exchanging data between applications or domains in highly distributed environment including ITS.

2.4.3. Semantic Agent

Semantic agents can be described as any type of intelligent agent which also exhibits or uses the semantics for system wide communication. The concept originated from the advent of semantic web however it is the least researched area in Agent-oriented computing.

Key to all agent interaction languages is a shared syntax and semantics of the domain (in the form of ontology) and the interaction protocols. In large distributed networks (e.g. enterprise intranets or large ITS systems) no single ontology can describe all the information. The vast majority of inter/intranet information is usually unstructured, informally interlinked and lacks formal semantics. These facts hinder agent-based activities over such a network and therefore, semantic agents have an edge and tend to be specialised for particular domains; i.e. those domains providing well-structured well-specified ontologies (Anon., 2009). Semantic web services provide an ideal medium for semantic agents' communication in form of service interface between agents using different level of ontologies.

In this research, the “control on demand” ability in agents design was one of the main motivations for using Agent-based Controls. The ITS@CU platform utilises Control Agents hosted on traffic controller devices and systems. These Agent-based controls extends the concept of Agents and fixed controls to another level by dynamically adding additional functionalities and decision making capabilities to the traffic systems/controllers. Control Agent can govern the operations of geographically distributed traffic controllers/systems over a network, and they also interact with other Agents on behalf of traffic Controllers in response to traffic situations and events.

Furthermore, the concept of semantic agents was combined with the Agent-based control concept in a novel way and hence in this research they are referred to as “Semantic Agent-based Controls”. The service agents are designed to integrate and communicate with semantic web services using the ontologies embedded within the service description. In this way the agents can find suitable services and interact with other agents in a more dynamic manner using flexible ontologies. The design details are covered in *chapter 4, 5 and 6*, and the implementation is covered in *chapter 7*.

2.5. Service Oriented Architecture and Grid computing

2.5.1. SOA Overview

The development of large ITS based systems are a highly complex process and require good software design, development and management processes. An efficient design and architecture plays a vital role in the success of such systems therefore adopting the right software architecture and programming paradigm becomes crucial. There are several software architectures such as:

- Blackboard
- Client-Server
- Distributed computing
- Front-end and back-end
- Monolithic application
- Peer-to-peer (P2P)
- Component Oriented Architecture
- Service Oriented Architecture
- N-Tier model

Most systems utilise multiple architectures, especially distributed systems which involve multiple sub-systems or components. Each architecture model has its limitation and benefits, and are suitable accordingly for example the Client-Server model is a good approach where multiple client applications require access to a central server but P2P does not necessarily require any centralised server. Individual components/sub-systems can have a completely different architecture relevant to its requirements; however, a system on the whole must have an architecture design which allows all the components and sub-systems to work in an efficient and integrated manner. Here SOA has an edge over all other architectures due to its interoperability and services based interfacing capabilities (Lorenz, 2006; Oracle, 2010).

SOA represents a system architecture which is based on a collection of services integrated in a loosely coupled manner. SOA is really about designing and building systems using heterogeneous network addressable software components in the form of Services (Lee et al., 2006). It is an architecture comprising components and interconnections that stress interoperability and location transparency (Strahle et al., 2007).

A general misconception is that SOA is just the use Web Services for integrating systems; SOA may be realised via Web services but Web services are not necessarily required to implement SOA (MSDN, 2007).

Properties of SOA

Use-based published contract: Contract design between components is a critical activity in SOA and it provides a published interface expressing the capabilities and purpose of the service.

Network Addressable Interface: A service must have a network addressable and identifiable interface and able to be invoked across a network.

Payload Format: The format of the data that is transmitted from a client to a service has to be commonly understood. For that reason, XML is the most popular format for data exchange in a SOA context.

Transport Type: The transport type or the protocol that is used to transfer data from client to service is usually HTTP/S and it is also used with web services to transfer service requests. The benefit of HTTP/S is that requests to a service are not blocked by the majority of firewalls hence it is considered as the most interoperable protocol. However, other types of transport types can be used for service invocation.

Interoperability: SOA stresses better interoperability and services must provide an interface that can be accessed across the network using a payload format and protocol that is understood by all of the potential clients of the service.

Discovery and Lookup: A service must be dynamically discoverable for the clients using directories which store information about the service. The directory information includes the service location, service operations and how these operations are invoked including the parameters and types that must be passed.

Main benefits of a SOA

The main benefits of a Service-Oriented Architecture are listed below:

- Reusable components or services
- Code mobility and rapid development of applications
- Standardised process & technologies
- Better scalability and transition to a responsive, flexible & extensible infrastructure
- More security and high availability
- Support for multiple client types
- Better maintainability
- Better testing hence less defects
- Ability to support cross functional and cross divisional processes
- Single implementation & enterprise-view of business services

- Service granularity recognised by a business user
- Cost reduction in longer term

2.5.2. SOA principles and architecture

SOA is based on various principles which provide a foundation for designing and developing SOA based systems and platforms. The *following are some of the core principles*:

- Compliance to standards (enterprise & industry)
- Loose coupling of services to minimise dependencies
- Service autonomy where services have control over the logic they encapsulate
- Service abstraction where services hide the internal functions/logic
- Service contracts are agreed and defined according the service-description
- Access disparate data via a single consistent access point
- Separation of business logic from the underlying technology
- Services are reusable and logic is divided into basic function services. Services are effective composition participants and multiple services can be used to provide composite functionalities
- Support cross-functional processes and service granularity
- Dynamic, discoverable, metadata-driven processes & services

Service Oriented design and architecture provides integration on a much higher level as shown in *Figure 2.7*. Object oriented design focuses on class level granularity and is tightly coupled to the system. In contrast, the SOA paradigm attempts to promote flexibility and agility through *loose coupling*.

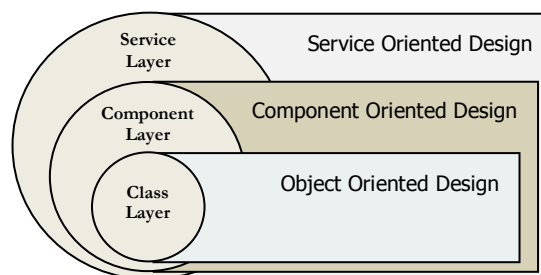


Figure 2.7: The layers of design (source adapted from: (Lorenz, 2006))

SOA services have two key roles: A *service consumer* and a *service provider*. The consumer client application invokes the services by sending request messages and then processes the response messages.

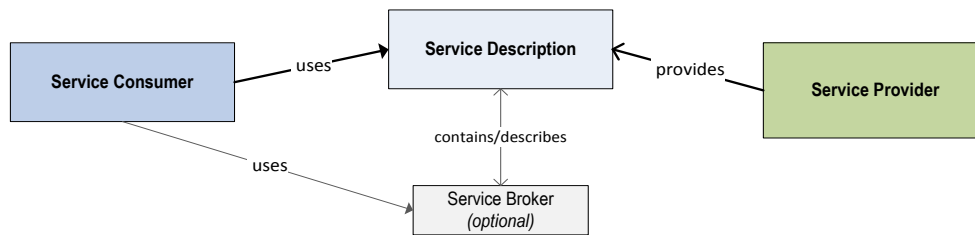


Figure 2.8: Service roles

The key terms that define the service in SOA are:

Service Contract: It specifies the rules of engagement between consumers and provider of the service. It is driven by business needs and specifies the policies that should be enforced and monitored.

Service Interface: It provides the consumers of a service to access its functionality according to its contract. A service implementation may have multiple interfaces or an interface can apply to multiple implementations.

Service Implementation: The implementation is the actual functionality used by the service, and may be accomplished using any technology.

Usage agreement: It is a contract between a service and its consumer derived from the overall contract to define the performance provided by the service and the level of usage.

A service consists of one implementation of one contract with one or more interfaces; and one or more usage agreements (Oracle, 2010)

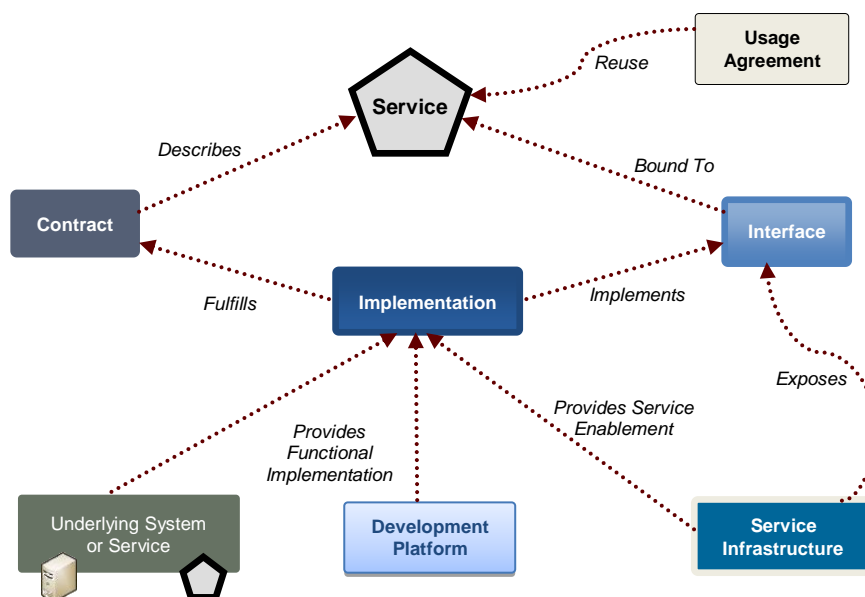


Figure 2.9: Service Definition

SOA Reference Architecture

SOA based architecture design is a complex process, and consistency, standardisation and best practices are critical. Every implementation of SOA should have a customised “Reference Architecture” to provide a framework that helps and guides the overall SOA implementation. Reference Architecture is the architectural blueprint describing all the services, its categories, support model, technical aspects, integration and underlying infrastructure and relationship with existing architectures. It plays a crucial role in SOA based system/solution implementations and serves as a communication vehicle, compliance tool, vision/roadmap and authoritative definition of SOA for the implementing organisation and stakeholders (developers, suppliers, integration partners etc.). A Reference Architecture must be based on well-defined SOA principles and best practices (Oracle, 2010).

SOA architecture is a layered approach to system design and integration. The following diagram outlines the basic layers used for SOA architecture.

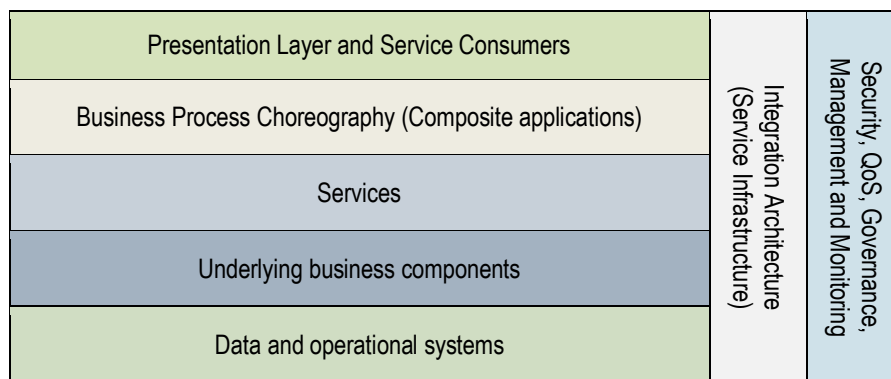


Figure 2.10: The layers of a SOA

Access or presentation layer: This layer provides access to service consumers such as Web Apps, Portals or client applications using browsers, PDA/Mobile etc.

Business process choreography layer: In this layer, services are bundled into process flows through orchestration or choreography, which act together as a single or composite application.

Services layer: Comprises of core services that are part of the entire SOA infrastructure. These services can be internal or exposed to external application or services. The services can be discovered dynamically and then invoked or choreographed into a composite service. Usually a Service Bus is used to provide all the service access, usage and orchestration features. Services are interpreted based on their service description and the term of use is based on the policy and service contracts. These services interact with other services and underlying business components and data access components. These services can exist in isolation or as a composite service.

Business components layer: Consists of components that are responsible for realising functionality and maintaining the Quality of Service (QoS) of the services. This layer typically uses container based technologies to implement the workload management, identity and authentication, high availability and load balancing.

Data and Operational systems layer: This consists of access/integration layer to databases, files and existing custom built or legacy applications.

Additionally, various supporting layers form part of a SOA infrastructure such as Integration layer, QoS, Service enablement, management/BAM, Governance and Security. Integration architecture generally refers to the Enterprise Service Bus (ESB) that supports and provides all the service access, routing, mediation, and translation of services, components, and their work flows. The services must be monitored and managed for quality of service and adherence to the non-functional requirements of the service (Arsanjani, 2004).

2.5.3. SOA related technologies

SOA is a technology independent architecture and the implementation can be accomplished using a wide range of technologies. *Some of the established technologies used for SOA implementation include:*

- Simple Object Access Protocol(SOAP) based Web Services
- Representational State Transfer (REST) or RESTful Web Service
- Microsoft Windows Communication Foundation (WCF)

Some older technologies can be also used in SOA implementation such as Remote Procedure Call (RPC), Remote Method Invocation, .NET Remoting, Distributed Component Object Model (DCOM)/DCOM+, and Common Object Request Broker Architecture (CORBA).

There are various vendors in the software industry which provide different development tools and technologies for SOA based implementation. *Some of the leading providers' offerings include:*

- **Oracle SOA Suite 11g:** It is an integrated suite of products to design, assemble, deploy and manage adaptable SOA based applications. It is based on industry standards targeted for corporate and large scale SOA and grid computing infrastructure systems. It includes products such as Oracle JDeveloper, Oracle Business Rules, Oracle BPEL Process Manager, Oracle Business-to-Business Integration, Oracle Business Activity Monitoring, Oracle Service Bus and Oracle Complex Event Processing.

- **IBM SOA Smart and SOA foundation:** It is a set of integrated software products and based on open standards and SOA's best practices and patterns. It is scalable, interoperable and allows a modular development approach. The software supports each stage of the SOA life cycle/stages i.e. model, assemble, deploy and manage. Some of the key products include WebSphere Business Modeler, Rational Software Architect, WebSphere Integration Developer, Rational Application Developer, WebSphere Portlet Factory, Rational Tester for SOA Quality, WebSphere DataPower SOA Appliances, WebSphere Process Server, WebSphere ESB, WebSphere Message Broker, WebSphere Portal, WebSphere Business Services Fabric, WebSphere MQ, Tivoli Access Manager, Tivoli Composite Application Manager for SOA, and WebSphere Service Registry and Repository.
- **Other vendors'** offerings include Microsoft Connected Services Framework (MCSF) and SAP Enterprise Services Architecture.

2.5.4. Critical analysis and drawbacks of SOA

Service Oriented Architecture is not always the best architecture for all type of systems because an optimal utilisation of SOA requires additional development and design attempts and supporting infrastructure which escalates the implementation costs (Exforsys Inc, 2007; Daigneau, 2011). Applications in homogenous environments, standalone, non-distributed, short lived and limited scope applications are not usually suitable for SOA (Altman, 2008; Lorenz, 2006; Exforsys Inc, 2007).

The governance of large SOA based systems is not an easy process and requires dedicated resources for service auditing, monitoring, versioning and change management problems associated with a large number of services (Oracle, 2010; Erl et al., 2012).

SOA has limited standards and every reference architecture of a SOA based system implementation is vastly different (Oracle, 2010; Daigneau, 2011). Although standardisation for SOA security, integration and management is progressing, there is still a long way to go before a coherent set of interoperability standards fully evolves (Exforsys Inc, 2007; Daigneau, 2011).

SOA is essentially based on a network centric approach i.e. it requires internet or intranet based network connectivity. This means that in highly distributed ITS environments where services and host systems are geographically dispersed, network latency issues are bound to have a negative impact on performance, especially in high network traffic or low bandwidth environments e.g. mobile/wireless communication (Schwab, 2007; Richardson & Ruby, 2007; Erl et al., 2012). SOA services lack proactivity features and there

are no adequate push-subscribe mechanisms without polling i.e. consumer clients (applications/services) usually invoke a service request and the service then responds.

Although SOA is not limited to web services (MSDN, 2007), web service technology forms a major building block of most SOA implementations. Web Services has its limitations and fails to achieve the goals of automation and interoperability because they require a well-defined service interface involving humans to write the description (McIlraith et al., 2001). WSDL specifies the functionality of the service only at a syntactic level allowing these descriptions to be automatically parsed and invoked by machines, however the interpretation of their meaning is still largely human (Martin et al., 2004).

SOA web services have additional overheads in terms of message payload, message processing, and connection setup and termination steps (Erl et al., 2012). SOA design process is complex and requires a considerable amount of experience in choosing appropriate platforms/technologies, types of web services (SOAP, RESTful etc.), payload formats (XML, JSON, Fast Infoset etc.), payload processing/parsing techniques and compression techniques (GZIP, SOAP extensions etc.). Most importantly the design should focus on reducing the number of service calls and effective service composition, orchestration, connection pooling and session management. Inappropriate choices often lead to inefficient SOA implementation, failure and cost implications (Lorenz, 2006; Daigneau, 2011).

The use of agents especially MAS in SOA is a relatively less explored area (Trullàs-Ledesma & Ribas-Xirgo, 2009; Gonçalves et al., 2009) hence it poses its own challenges due to the lack of agent tools and support in SOA. Using Agents can add more complexity to services invocation especially for asynchronous service calls resulting in inconsistent service flows and difficult to track service activities. The use of agents across service bus and the self-organisation of service flows (orchestration and composition) is also challenging with agents. Agents can fight over the service access leading to conflicts which can result in a loss of service response or delays (Ribeiro et al., 2008; Ricci et al., 2007). All such issues must be considered and addressed when implementing SOA based systems using agents. The next section further discusses the adoption of SOA for the agent-based ITS platform in this research.

2.5.5. SOA in the context of this research

Agents in geographically distributed and heterogeneous environments (such as ITS) have implementation difficulties due to interoperability among agents, and therefore require a unified software platform and standard implementation protocols (Dasheng, 2010; Ricci, 2007). SOA is based on a unified standard, and it is platform-independent, loosely coupled and provides an ease of integration which enables system interoperability and computing resources reuse. Web services can also embed agents or use agents to send and receive messages, while the services themselves are the resources characterised by the functionality it provides (Luck & McBurney, 2008). The combination of MAS and SOA is a promising approach to address the

interoperability, unified communication and other such complexity issues associated with modern ITS systems. SOA provides an ideal underlying framework to implement MAS in distributed and heterogeneous ITS environments (Dasheng, 2010; Tapia, 2008; Booth et al, 2004).

The overall architecture of the platform in this research is based on SOA which provided seamless integration required for distributed traffic control systems and Agents from multiple domains (Wang et al., 2010). A customised “Reference Architecture” was designed as a framework to help and guide the overall SOA implementation of the platform. The Agent communication layer was developed using a combination of SOAP and REST services specifically to allow multi-agent communication in the SOA environment.

Additionally, the platform supports ontology based semantic web services which are not limited by the requirement of a well-defined service interface of conventional web services. A WSDL specifies the functionality of the service at a syntactic level, allowing these descriptions to be automatically parsed and invoked by machines. However the interpretation of their meaning is still human based. Semantic Web services address the limitations of current Web service technology by augmenting the service descriptions (through the addition of ontologies/semantic metadata to the information resources) with a semantic layer in order to achieve automatic discovery, composition, monitoring and execution (Li et al., 2005; Sabou, 2006; Altova, 2007). *Chapter 7, sections 7.2 and 7.3* discuss the implementation details of the SOA reference model and the Agent messaging and communication layer in this research.

2.5.6. Grid computing

Grid is a distributed system of networked computers (with no or limited central control) cooperating and sharing resources to perform tasks more efficiently, and to minimise the processing idle time. Grid technology also provides an ability to store, share and analyse large volumes of data, improves resource utilisation, efficient access to information which can improve decision making, productivity, collaboration and flexibility of the overall system. *Grid computing can be divided into three stages based on their sizes:*

Cluster Grids: Consist of several systems connected together (forming a cluster), providing a single point of access to users, and perform shared tasks, and share the workload and resources. Such grids are typically owned and used by a small number of users e.g. a project or department level teams.

Campus/Enterprise Grids: Consist of multiple cluster grids to achieve computational outcome on a wider scale. Mainly used by large organisations with distributed networks dispersed in different locations to cooperatively share the computing resources across the enterprise.

Global Grids: This level includes the geographically dispersed grids of enterprise and cluster levels combined to form larger global grids. Primarily designed to support and address the needs of multiple sites and organisations sharing resources and workload. The participating grids agree upon global usage policies and protocols, however not necessarily the same implementation.

Semantic Grid

This is a grid computing approach using the semantic data model to describe the information, resources and services. It is based on the concept of semantic web, and can be defined as an extension of the current grid approach where the services have well defined meaning in form of ontologies. It enables easier discovery and automatic join up of resources, which helps bringing the resources together much more efficiently over the distributed network to form virtual organisations.

Grid computing in ITS

The problems of large scale ITS based infrastructure includes:

- Integration of heterogeneous data,
- Management of dynamic service flow,
- Cooperation among different domains, and
- Store and share massive data among different systems and departments

Grid computing offers a great opportunity to build a synthetic platform to address these issues. It can support traffic data semantisation, resource sharing, ITS subsystem cooperation, and distributed computing to connect all kinds of ITS systems and resources (Wu et al. 2005). Therefore, Grid based distributed architectures are ideal in an ITS context, to share the load and resources and minimise the role of central server systems.

This research utilises the Grid based approach in the ITS@CU platform by dividing traffic network infrastructure/systems into smaller areas (grids), which are controlled by autonomous subsystems (local controllers). The local controller governs the ITS elements within its grid area, therefore reducing the load on the central server. It also increases the responsiveness of the system by allowing local controllers to make decisions independently at the local/grid level and share resource utilisation. Additionally, a semantic layer is added to the grid approach allowing the services, agent controls and traffic controls to communicate at a cross domain level.

2.6. Review of the Related Work

This section reviews some of the relevant research work with similarities either at the approach or component level. In addition, it reviews available technologies, platforms and approaches at the time the research was carried out in order to justify the approach adopted for ITS@CU. There are currently no platforms available which, as a whole, can be directly compared with the ITS@CU implementation approach due to the unique functionality and combination of technologies/concepts employed in the approach.

2.6.1. Related research platforms and systems

Over the years, various research initiatives and approaches have been adopted for the efficient management of traffic such as distributed systems, Multi-Agent networks, artificial intelligence, artificial neural network, Grid Computing and Telematics. The automation of the ITS infrastructure using Agent-based Controls, and integration using semantic services and SOA principles have to date received very limited attention (Dasheng et al., 2010).

The following table outlines some of the platforms and systems, and its relevance to this research:

System/ platform	Description and analysis
DATAGRID II (Wu et al., 2005)	It is a semantic ITS platform and uses the concept of grid technology and allows resources sharing and multi-platform service flow and cooperation. This system has some similarities with the ITS@CU platform as it also has a layered architecture approach and the components of the systems are grouped into different layers. However it lacks the concept of SOA and Agent based controls which are the key differentiators.
FUSION@ (Tapia et al., 2008)	FUSION stands for Flexible User and Services Oriented multi-ageNt Architecture. It is a multi-agent architecture which facilitates the integration of distributed services and applications to optimise the development of agent systems. The core of the architecture is a group of deliberative agents acting as controllers and administrators for all applications and services. The functionalities of the agents are not inside their structure, but modelled as services (Tapia et al., 2008). This system approached the SOA based

	<p>communication strategy where applications and services can communicate in a distributed way independent of a specific platform.</p> <p>The system has some similarities with the way ITS@CU platform Agents are designed. However, the focus of this system is on de-centralisation as compared to the hybrid approach in ITS@CU. Also the services in FUSION@ are atomic/single service which complicates the use of multiple services as composite workflows i.e. a client application has to perform the composition of multiple services which adds processing overhead. FUSION@ supports a limited number of Agents i.e. controllers and coordinators. Currently, the system is in its early research stages and very limited test results have been published.</p>
<p>MADARP</p> <p>(Cubillos et al., 2005)</p>	<p>MADARP, Multi-Agent Architecture for Passenger Transportation Systems, is a multi-agent architecture for passenger transportation systems. The architecture provides agents that implement basic planning and control functionality to process transport requests coming from different users.</p> <p>It provides a set of base agents that perform the basic interface, planning and support services for managing different types of transportation requests by using a heterogeneous fleet. The agents allow adapting the architecture to different planning models by integrating vehicles and users in a pervasive way. It also includes three planning models implemented using the architecture. It provides an interesting approach to the use of Agent based technologies in ITS context.</p> <p>MADRAP has some similarities to this research in Agents organisation and some communication aspect; however its focus is mainly Passenger transport.</p>
<p>IITS UCSD</p> <p>(Trivedi, 2001)</p>	<p>ITS and Telematics research initiative at University of California at San Diego. It is an incident detection system and uses the basic concepts of Agents. It uses image processing techniques for analysing incident situations. The research also introduces mobile platforms in support of various spatial search or other interactions desired by a remotely stationed human in this environment.</p> <p>The mobile platform and the use of wireless based sensory controls offers some similarities in relation to this research, however the wireless communication technologies used are now out-dated and do not incorporate ad-hoc networking features as compared to ITS@CU.</p>

<p>WAIMSS</p> <p>(Ozbay & Kachroo, 1999)</p>	<p>Wide-Area Incident Management Support System (WAIMSS) is an incident management system implemented mainly in Java. It uses various incident prediction algorithms, rules and models. It includes GIS capabilities and various research studies have been carried out on this system.</p> <p>The rules in WAIMSS for detecting traffic abnormalities has a similar approach to this research, however the main difference is that ITS@CU uses Control Agents to analyse and select appropriate rules from a rule-set data dynamically in response to different traffic situations (detailed in <i>chapter 6</i>).</p>
<p>RHODES</p> <p>(Mirchandani & Wang, 2005)</p>	<p>RHODES (Real-Time Hierarchical Optimized Distributed Effective System) is an Advanced Traffic Management Systems (ATM) for intelligent transportation systems. RHODES takes sensor-based traffic data as input and outputs traffic signal timings to optimally control traffic flow.</p> <p>RHODES also uses a similar concept of a Control Algorithm which allows for controlling multiple controls over the network. RHODES limits the control algorithm to the intersection level (traffic control devices level), however ITS@CU uses Control Agents at traffic Controllers and Grid level. RHODES uses fixed control algorithms only for prediction and estimation of traffic flow but ITS@CU uses a dynamic approach and adds domain semantics allowing Control Agents to expand its functionalities.</p>
<p>SOA based VANET</p> <p>(Goncalves et al., 2009)</p>	<p>This is a SOA based framework for Vehicle to Vehicle communication. It couples the SOA concepts with Vehicular Ad-hoc NETWORKS (VANET), where each vehicle communicates with other vehicles using a services layer. It forms a mesh of ad-hoc networked vehicles within a limited range and each vehicle becomes a node. These nodes provide (publish) their functionalities in the form of services for other nodes to use (consume).</p> <p>The simulation framework has some similarities to the V2V SOA coupling in ITS@CU. The framework is limited to V2V communication and relies on mobile wireless (GPRS) technologies and GPS location data to form ad-hoc network between vehicles within a set range. In comparison, ITS@CU uses Bluetooth and IrDA for ad-hoc network formation which means it is not dependent on mobile network (3G/GPRS) coverage or GPS to form an ad-hoc network between vehicles within the range of each other.</p>

Table 2.5: Related ITS research systems

2.6.2. Relevant Agent oriented approaches and studies

Various Agent oriented approaches have been applied in recent years mainly to traffic modelling and simulation, dynamic routing and management of intelligent traffic control and congestion.

The following table outlines some of the Agent-based approaches and their applications:

Approach/ Systems	Description and analysis
aDAPTS (Wang, 2005)	<p>aDAPTS is an Agent based networked traffic management system. It proposes the use of Agent-based controls which decompose a sophisticated control algorithm into task oriented agents distributed over a network. The ability of dynamically deploying and replacing control agents as needed allows the network to operate in a control on demand manner to adapt to different control scenarios.</p> <p>According to Wang (2005), the system architecture employs a three level hierarchical architecture. The highest level performs reasoning and planning of task sequences for control agents; the middle level dispatches and coordinates control agents; and the lowest level hosts and runs control agents. The control agents are represented by mobile agents that could migrate from remote traffic control centres to field traffic devices or from one field device to another.</p> <p>This is one of the very few research approaches which uses Agent-based controls in ITS. As compare to aDAPTS, the ITS@CU platform adds a semantic layer to the Agent-based controls and the agent can split its functionality on different hosts which gives it an advantage especially in limited resource hosts. Another key differentiator is that the ITS@CU platform is based on SOA principles as compared to aDAPTS architecture which is tightly coupled and difficult to integrate.</p>

<p>TRACK-R</p> <p>(Garcia-Serrano & Vioque, 2003)</p>	<p>Traffic Agent City for Knowledge-based Recommendation (TRACK-R) is a roadway traffic detection and management system. It is compliant with the IEEE Foundation for Intelligent Physical Agents (FIPA) standards.</p> <p>TRACK-R comprises of agents responsible for a geographical area and they provide traffic route recommendations for humans or other agents. The agents in these two systems are implemented using the JADE agent platform.</p> <p>The ITS@CU platform also includes Service Agents to provide route guidance however the approach is different. The route guidance agents work with control agents and can be configured to provide route diversion information which can be used by multiple control agents to adapt new routes, for example change the green time on traffic signals and dynamic diversion messages on the affected route to inform approaching vehicles.</p>
<p>Mobile-C</p> <p>(Chen et al., 2006)</p>	<p>Mobile-C is an IEEE FIPA standard compliant multi-agent platform. It supports both mobile and stationary agents in networked and embedded systems. It is based on hybrid control architecture and uses C/C++. Mobile is an academic level platform for agent development. As compared to the ITS@CU platform it is limited and does not support SOA and other complex requirements of this research such as .NET, SQL Server and mobile application integration support.</p>
<p>TRYSA2 and InTRYs</p> <p>(Hernandez et al., 2002)</p>	<p>InTRYs and TRYSA2 are agent-based intelligent traffic management systems. InTRYs achieves agent coordination based on a traditional centralised mechanism, whereas TRYSA2 employs decentralised coordination. The decentralised architecture has advantages in terms of synchronisation, reusability, and scalability. However, regarding the complexity of the coordination task, the InTRYs approach is better than the decentralised system TRYSA2. This is because the TRYSA2 strategy may apply an exhaustive search for plans to be selected by the involved agents.</p> <p>ITS@CU uses a hybrid approach as the above study shows both centralisation and decentralisation have different advantages and weaknesses. A fully centralised approach increases delays in decision making as the central control requires communication with all the controls involved to make a decision. Conversely fully decentralising the system to agent and grid control level requires devices (e.g. traffic light) with more processing power and memory. The ITS@CU platform follows a hybrid approach where the central system arbitrates on a higher level and provides essential services whilst Agent-based controls perform</p>

	tasks in association and cooperation with each other (and utilising shared resources) for local level decision making and operations.
CARTESIUS (Logi & Ritchie, 2002)	<p>This is an Inter-jurisdictional traffic congestion management system for motorway/freeway networks. It is composed of two decision support agents 1) A motorway/freeway agent which analyses congestion and 2) An arterial agent which generates appropriate responses to the traffic situations. Both agents interact in real-time to support incident management operations and interact with human operators to determine control recommendations in response to incidents. The agents continuously receive real-time traffic, incident detection and control status data, and analyse the data for congestion or a potential incident. In the event of an incident, it formulates solution recommendations to deal with the incidents. These recommendations appear on the user interface of the application to allow operators to agree and select a solution.</p> <p>ITS@CU uses a similar approach to monitor the traffic flow and Agents interact with other Agents in a similar fashion. However, the agents involved in ITS@CU have a different design structure and organisation hierarchy with more functionalities and services which provides an advantage in dealing complex scenarios.</p>
(Roozemonnd, 2001)	<p>Roozemonnd suggested a Cooperative Traffic Signals approach using an agent-based urban intersection control system that reacts to changes in the traffic environment and adapts itself to changing environments based on internal rules. It comprises various agents such as Intersection Traffic Signalling Agents (ITSAs), Road Segment Agents (RSAs), and some authority agents. All the agents collaborate with each other to manage the intersection controls.</p> <p>The behaviour of traffic lights adaptation using Agents has some resemblance to the approach in ITS@CU however it lacks the ability to share the same data between multiple intersections such that each intersection can get an accurate picture of the current traffic situation.</p>
(Srinivasan & Choy, 2006)	<p>This is a multi-Agent system comprising of different levels of Agents; the lowest layer uses an Intersection Controller Agents (ICAs); the middle layer uses a Zone Controller Agents (ZCAs); and the highest layer uses a Regional Controller Agents (RCAs). ZCA controls several pre-assigned ICAs, and one RCA controls all of the ZCAs. The implementation of agents is based on neural</p>

	<p>network and fuzzy logic theories. It allows agents to dynamically adapt to the changing environment using reinforcement learning, weight adjustment and dynamic update of fuzzy relations using evolutionary algorithms. The architecture has some similarities in terms of agent organisation but the design approach is completely different.</p>
<p>Platoons (Clement & Taylor, 2006)</p>	<p>Clement and Taylor focused on creating platoons of vehicles in order to minimise the effects of stop-and-go driving using a model called "Simple Platoon Advancement" (SPA). ITS@CU does not use platooning algorithms however it attempts to automate intersections by adapting traffic lights/signals using Traffic Light Controls Agents.</p>
<p>(Balan & Luke, 2006)</p>	<p>This is an intersection control method using a history based approach to maximise fairness (all vehicles experience similar delays) as opposed to efficiency (the average vehicle experiences short delays). Vehicles which have historically experienced long delays should be more likely to experience shorter delays at subsequent intersections. In addition to being a multi-intersection approach, this method uses a marketplace model involving a system of credits that can be given and taken in exchange for shorter and longer delays, respectively. Coordination at individual intersections is still done with traditional traffic lights, the timings of which are part of the mechanism. Interestingly, the fairness approach actually yields results that are also reasonably efficient.</p> <p>The concept of system credits provided an efficient method to moderately distribute the traffic load therefore it was also explored for the ITS@CU platform and can be configured for Agents to use system credits for routes affected by diversions. So if any given route is affected for a longer time then the system credits can be increased for the route, which are configurable values (set by the administrators).</p>
<p>(Balbo & Pinson, 2005)</p>	<p>This is a multi-agent based Distributed Decision Support System for urban public transportation system management. It introduces an interaction model called Environment as Active Communication support (ESAC) which allows agent interaction by sending and receiving messages through logical filters of emission, reception, and interception. The communication approach shares similar methods with ITS@CU for filtering messages for example ontology selection process by control agents during communication with Service Agents.</p>

Table 2.6: Relevant Agent Oriented approaches and studies

2.6.3. Related commercial traffic management systems

The author worked with the UTM Control Room, Coventry City Council responsible for managing the traffic of Coventry City and surrounding areas. They use SCOOT and related add-on systems for managing traffic control systems. They provided an extensive range of real historical data for different routes of Coventry City for the purpose this research. A summary report of the collaboration work is covered in “Appendix L”.

There are various other commercial systems for managing traffic control systems however SCOOT and associated add-on systems are widely used. The following table outlines these systems and their relation to the research:

System	Description
SCOOT	<p>SCOOT, the Split, Cycle, and Offset Optimisation Technique, is an urban adaptive Traffic Control system for optimising traffic signal timings in a network to minimise delays and stops. SCOOT is an on-line adaptive traffic control system that can react to changes in traffic flow, give priority to public vehicles such as buses, and estimates vehicle emissions. While SCOOT has been shown to offer a better performance in reducing traffic delays by an average of 20% over systems like TRANSYT, SCOOT requires reliable traffic data in order to adapt, and thus may be slow in its reaction to changes in traffic flow (Siemens, 1999).</p> <p>SCOOT is widely used across the world and therefore in this research the author has studied SCOOT and its associated systems such as ASTRID and INGRID. As part of this process, the author used the data provided by Coventry City Council Communication Centre to analyse traffic trends on some of the routes in Coventry City centre. The data was also used in simulation studies and imported into the ITS@CU platform.</p>
ASTRID	<p>ASTRID, Automatic SCOOT Traffic Information Database, is a database designed to collect information from a SCOOT traffic control system, or other source of time-varying traffic data. ASTRID automatically collects, stores and processes traffic information (such as delays, flows and congestion) used by the SCOOT model in the optimisation process. ASTRID is generally used for display or analysis of the SCOOT data by engineers and researchers. If the data is converted into an appropriate format, ASTRID can also process</p>

	<p>data from sources other than SCOOT. (Siemens, 1999).</p> <p>The author used mainly ASTRID system for SCOOT's historical data extraction and detailed analysis of selected traffic corridors to and from Coventry City Centre.</p>
INGRID	<p>INGRID is a real-time traffic incident detection system using information from SCOOT. Once an incident has been detected, information on the location and severity of the incident can be passed to the traffic operator. INGRID uses an algorithm to detect traffic incidents by examining the current traffic data for sudden changes in detector flow and detector occupancy (SCOOT-UTC, n.d.).</p> <p>In this research, INGRID algorithms were also studied to develop the Control Agents incident detection capabilities. INGRID uses information from connected detectors on roads and detects incident location by analysing detectors at a point where an upstream detector shows a queue of vehicles (i.e. occupancy greater or flow lower than the normal smooth value over recent minutes) and a downstream detector shows the opposite level of flow and occupancy. This approach was adopted in the ITS@CU platform and the Agents collect vehicle count sensor information in a similar fashion and then report the occupancy and flow abnormalities to an operational Agent for further analysis and actions to restore traffic flow. The evaluation <i>chapter 8</i> presents a congestion detection study detailing this approach.</p>

Table 2.7: Related commercial traffic management systems

Remark: A further review of the relevant simulation systems is included in “*Appendix C, section 7*” and relevant Agent tools and technologies are covered in “*Appendix D, section 2*”.

2.7. Conclusion

This chapter has reviewed the theoretical and technological aspects of various areas relevant to this research. It has been discussed that an approach using a combination of multi-agent, SOA and semantic web services provides an alternative approach to implement cooperative and efficient ITS systems with the capability to self-organise by responding to emergent traffic situations. Agents can be used for controlling the distributed traffic systems/devices over the network; however these systems and controls are usually multi-domain and technologically diverse. SOA provides ideal underlying architectural principles for seamless integration and communication between these agent-enabled traffic controls/systems. Furthermore, the SOA enabled agent communication can be enhanced by using semantic web services concepts allowing easier service discovery.

The final part of the chapter presented a review of related work, researches, studies and commercial systems which supported the need for an alternative approach and justified the investigation into semantic agent-based controls. The lack of ITS platform availability, especially on a commercial level, indicated the need for the development of a new platform to achieve the main research objectives.

Chapter 3

Road Network Model Development

The previous chapter reviewed the theoretical and technological aspects of various areas relevant to this research such as ITS, Agent oriented technologies, Semantic Agent based Controls and SOA. It also presented the review of various research systems and studies related to this research.

This chapter is focused on a road network model which is adapted and used as a foundation for the design of the Agent-based Controls and the ITS@CU platform components in this research. The chapter has the following sections:

- The first section provides an overview of the model and states why a new road network model was required
- The second and third sections outlines the elements of a typical road network and their mapping strategies
- The fourth section describes the road network model, attributes of its elements and their relationships.

3.1. Overview

Several traffic/road network models for ITS have been developed in the past few years ranging from a generic traffic flow models to road network classification models to infrastructure specific models. During the research, the author assessed various existing models both of microscopic and macroscopic type (discussed in *chapter 2*); however these models were not directly applicable to the agent-based controls approach in the ITS@CU platform and its design specification. Therefore, a traffic/road network model was developed specifically adapted for distributed agent-based controls and to efficiently utilise semantic services. This model is based on SCOOT (Siemens Mobility, 2009), however with the following additions:

- Support for the grid-based traffic network
- Support for the Agent-based Controls
- Simplified data representation of the model elements and implementation on the database level to support scalability of the platform (important for SOA)

Traffic networks are of different types such as road network, rail network, waterway network and air traffic network. This model is specific to a road based network.

3.2. Road network elements

A typical road network consists of the following main elements:

Road: A road is the main element of the traffic network, which forms a link or path connecting two junctions or intersections. There are different types of roads such as one/two way, single/dual carriageway, motorway, slip road and so on. A road has various attributes and properties such as name, type, number of lanes, length, speed limits and location.

Junction: A Junction is a traffic element which binds roads and represents a physical connection between its adjoining roads. A junction has different types such as box junction, T junction, roundabouts, and crossings. Junctions always have some form of traffic flow control to avoid accidents e.g. traffic lights, signs or road marking.

Traffic areas and points: The traffic points or areas refer to places on the road network designated for traffic related purposes e.g. bus stops, parking on road side, tolls, services.

Beside the aforementioned main elements, the road network also contains the following infrastructure controls:

Traffic flow controls: The flow of traffic is always controlled by various flow control elements such as traffic lights, road/lane marking (give way, zebra crossing, no stop, turns) and signs (speed, stop, prohibited vehicle type, prohibited turns and one way). The flow controls are mainly located at intersections or junctions to avoid the incidents and manage the traffic stream.

Traffic Information: On road information and warning for drivers such as signposts, signs, road markings and variable message display boards.

Traffic monitoring and data collection: Devices and systems which monitor the traffic such as CCTV, camera capture (for speed and signal break), inductive loop sensors, toll collection, weather sensors and so on.

3.3. Road network mapping

Another aspect of road networks is mapping and routing, and the traffic networks (represented as digital maps) follows graph theory which states that a graph is a pair $G = (V, E)$ of sets where the elements of V are the vertices (nodes or points) of the graph G , the elements E are its edges (lines, or arcs, or link) (Xia et al., 2004). This means that a graph can represent a map as its main building blocks are two elements 1) a road which is simply an edge or line and 2) a junction/intersection which can be assimilated to a graph's nodes. Since roads always have a direction (two-way or one-way) a map can be represented as a directed graph. Most of the mapping related technologies such as GIS, GML and GDF (discussed in “Appendix C, section 8”) are based on the link-node approach.



Figure 3.1: Representation of map as a graph (Red arrow showing one way road)

3.4. Model description

The model developed takes into account all the main traffic elements and infrastructural controls, and classifies them into the following categories:

Grid: A typical traffic network infrastructure covers a vast area, and in some cases it covers an array of cities linked closely to each other. In order to manage the infrastructure, it is always divided into smaller zones or sections by the authorities and infrastructure developers. In this model, a “grid based” approach has been adopted as a way to divide the road network. Grids are virtual representations of geographic areas of the traffic network infrastructure as shown in *figure 3.2*. All systems and devices related to the traffic infrastructure (on-road/roadside), which are within the boundary of a grid, belong to that grid on a system level.

A grid covers an area based on a combination of both size and the number of systems within that grid. The grids do not overlap and have no gaps between them. While forming the grid the actual road network and road intersections are taken into account so that a grid always covers roads with as many of their interdependent intersections as possible (As seen in *figure 3.2*, a map with grid edges).



Figure 3.2: Map showing Grids in green colour boundary line (Coventry City Centre)

For the modelling purpose, the area around Coventry city centre was surveyed (see “*Appendix L*”) and used as an example for a typical urban traffic network.

A Grid has the following properties:

Grid Property	Description
Grid ID (G_i)	Each grid has a unique identifier to distinguish it from other grids
Boundary	Polygonal geocode/geometry (List of all points in longitude/latitude) representing its location and boundary
Type	Rural, urban or motorway

Table 3.1: Grid properties

The grid based distribution approach of the traffic infrastructure has many benefits.

- Highly scalable: Grids can be added and resized easily
- Easier management of the traffic infrastructure components
- Easier way of pinpointing the traffic problem
- Efficient data sharing and communication between grids in distributed environment

A Grid is always controlled by a “grid controller”, which is responsible for managing all the systems/devices/controls within the grid and for outside communication i.e. with control centre, vehicles and other grid controllers.

Node: A node is simply a point on a road network where the traffic flow stops or converges (e.g. a junction). It joins one or more inbound and outbound links. A node contains some form of flow control such as traffic lights or signs/markings to avoid collisions. A node in the context of this model is a single point, which consist of all traffic flow control points directly dependant on each other e.g. a pedestrian crossing on one side of the road which depends on the other side forms single node, or a junction or roundabout may have multiple traffic lights which are dependent on each other and must be coordinated with each other to avoid collision. In this model the nodes are divided in two types: passive and active, see *table 3.2* which defines the Nodes properties.

Node Property	Description
Node ID (N_i)	Unique identifier of the node
Type	Active or Passive
Boundary/Location	Polygonal geocode/geometry (List of all points in longitude/latitude) representing its location and boundary
Structure Type	Roundabout, Junction, Crossing etc.
Links-In	List of incoming links
Links-Out	List of outgoing links
Current Status (Only for active node)	The current status of each route combination (Link-In/Link-Out) within the node i.e. Busy, Normal
Request Timeout (Only for Active node)	The timeout period (in minutes) for a change in signal adaptation based on the Current Status of links within the node.

Table 3.2: Node properties

Active Nodes are traffic flow controls involving traffic lights i.e. they are controlled by traffic signalisation principles. They play an important role in managing and distributing the demand of the traffic flow. Active nodes change their behaviour based on the traffic condition and external systems' requests. For example they can increase/decrease the red/green signal time to accommodate for changes in traffic flow.

An active node resembles a black box i.e. a single control, exposing only relevant external attributes and functionalities, and encapsulates and hides its internal controls functionalities and properties. An active node on a system level is controlled by the node controller responsible for all the controls within the node such as traffic signalisation, message displays, and the coordination and synchronisation of the signals in opposite direction (or dependant).

A node as a single control is an important aspect of the model to reduce the complexities at the traffic network level i.e. by leaving each node controller to manage its internal controls and functionalities. Such decentralisation makes it an ideal candidate for agent-based controls. A node allows external systems/controls (Grid controller or other node controllers) to either enquire its current status (or other properties) or request permissible actions e.g. to allow extra traffic stream from certain incoming link(s) going to certain outgoing link(s). A node assesses its internal status and accepts/rejects any requests accordingly. If an action is requested and the node is in a position to perform that request, it uses its internal properties and functions e.g. if it needs to allow additional traffic from incoming link L_1 going towards L_3 (by increasing the green signal time on these links), and if L_1 depends on L_2 (L_2 red signal time must be increased to avoid accidents) then the node itself deals with all the internal adjustment/signal synchronisation and behaviour.

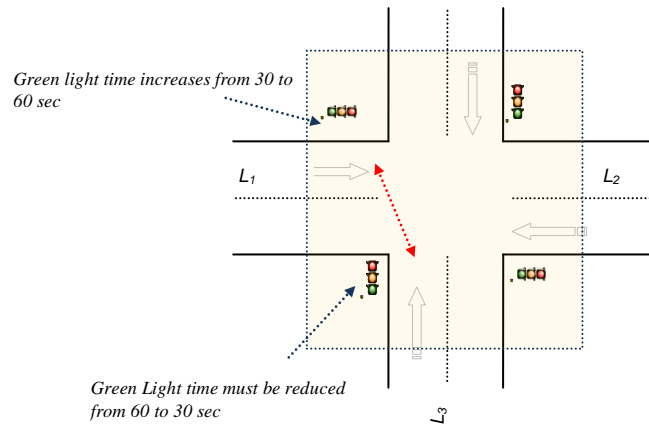


Figure 3.3: Internal signalisation adaptation

There are some other types of traffic points for example toll services or rail junctions which qualify as active nodes, however in this model such points are not considered as active nodes, as they are more rigid and signal adaptation is difficult or not possible in some cases.

Passive Node: This is a type of node point where the traffic flow is not controlled through signalisation but through general traffic rules such as road marking and signs (e.g. T-junctions, roundabouts). There are always more passive nodes than active nodes in any traffic network especially in residential built-up areas. A passive node does not have any internal controls and functionalities and it is only used for route formation.

Link: This is a connection between two nodes. It comprises multiple roads and is multi-directional i.e. upstream or downstream. Each stream can consist of multiple lanes.

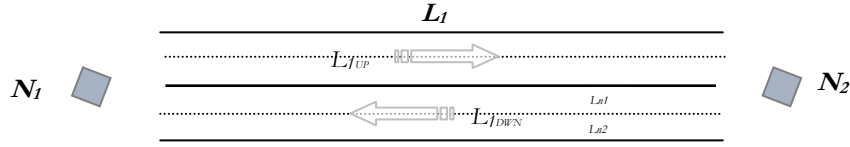


Figure 3.4: Link with up/down stream and lanes

A link has the following properties:

Link Property	Description
Link ID (L_i)	Each link has a unique identifier
Boundary/Location	Polygonal geocode/geometry (List of all points in longitude/latitude) representing its location and boundary
Length	Length in meters
Road name	Name of the road on which the link exists
Road Type Info	Type information of the road on which the link exist <i>e.g.</i> Motorway, Dual Carriage way, Side street, One-way.
Number of lanes	Lanes on the road
Speed	Default or normal speed
Max Speed	Maximum speed allowed
Max Capacity	Maximum number of vehicles allowed
Priority weighting factor	Not all links in a network are of equal importance. Each link has its priority in form of a weighting factor value
Current flow status	Current average speed and the number of vehicles

Table 3.3: Link properties

Route: An ordered list of links and nodes. It is connected by a start and end node (with no repeating nodes). Route R is a sequence of distinct nodes (N_1, N_2, \dots, N_n) such that $\langle N_i, N_{i+1} \rangle$ (or $\{N_i, N_{i+1}\}$), for each i from 0 to $n-1$, is a link in G . The path is *simple* if it does not cross itself and is a *cycle* if $N_0 = N_n$ i.e. the start and end node are same.

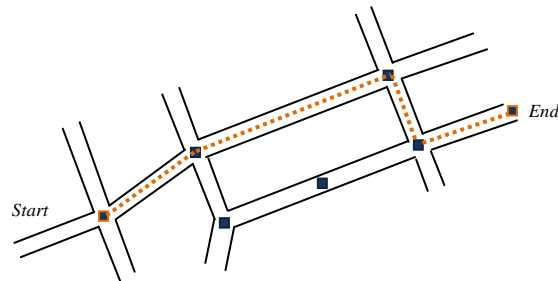


Figure 3.5: Route

Efficient route formation is important for optimising the traffic flow, diversion, route planning, navigation and managing the controls network flow. A route is formed using shortest path algorithms (such as Dijaskra and A*) which identify a set of specific simple paths or routes paths whose length is smaller than a given length threshold.

Controls (Traffic Infrastructure): Road network/traffic infrastructure consists of various control systems (roadside sensors, devices, traffic lights, message displays, vehicles etc.), and efficiently controlling and managing these controls is vital in ITS infrastructures. *In this model the controls are divided into the following types.*

- Flow control (Active nodes)
- Informative (Message boards, signs etc.)
- Monitoring (Traffic data collection sensors, CCTV etc.)

All controls are either part of a link or a node *e.g.* Message display is part of the road/link it is placed on. Other controls include a grid controller which is part of a grid network and vehicle controls which temporarily becomes a part of the grid where it is presently located.

A Control has the following properties:

Control Property	Description
Control ID (C_i)	Each control has a unique identifier
Location	Location in terms of longitude/latitude of the control
Type	Node Controller, Grid Controller, Message Board, Traffic Signal/Lights, Inductive Loop, Sensor, CCTV etc.
Part of	The ID of the node, link or grid the control exist or belong to
Current Status	Current status depending on the type <i>e.g.</i> Traffic light could be in default mode or High green mode, whether sensor to return current condition, node controller to return individual Links-In/Links-Out status etc.
Controlled by	ID of external controller or control Agent. Controls functionality can be controlled by Agents.
<i>Other control specific properties</i>	As controls are of different type they will have their own additional properties/attributes

Table 3.4: Controls properties

Vehicles: This is a dynamic type of control entity in this model. A vehicle is always part of the grid it is presently in. The vehicle may be complex in itself and may have several internal controls, however from the model perspective it is a single control. The vehicles are of different types such as standard, logistics, public transport or emergency. The vehicles properties are outlined in the following table.

Vehicle Property	Description
Vehicle ID (V_i)	Each control has a unique identifier
Location	Last known location in terms of longitude/latitude
Type	Standard, logistics, public transport, Emergency/Priority
Dimensions	The vehicle dimensions/size in term of its height, width and length
Current Status	Idle, On the move etc.

Table 3.5: Vehicle properties

All the elements of the model are related to each other and on a system data level they form the relationships illustrated in *figure 3.6*.

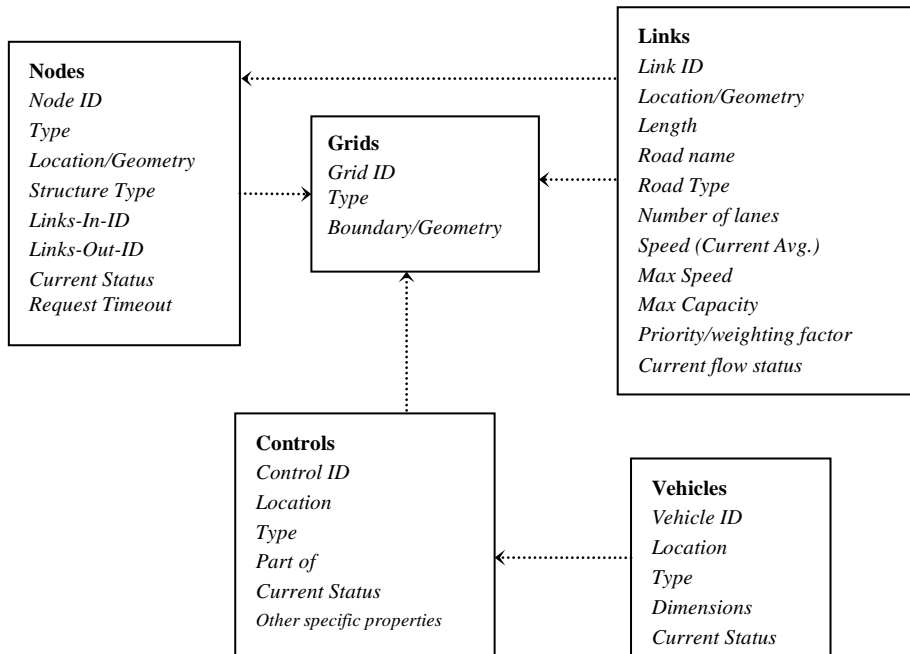


Figure 3.6: Elements Data Relationship Model

3.5. Conclusion

This chapter has described the new customised road network model, the attributes of its elements and their relationships. It provides the foundation for describing the Agent-based controls and their data level relationships in the ITS@CU platform. The model is also important for the design of the platform and elements of the model form the basis for all the components/sub-components of the platform.

Chapter 4

Agent-based Controls Design & Organisation Structure

The previous chapter presented the road network model which is used as a foundation for the design of the ITS@CU platform components and the Agent-based Controls described in the next section.

This chapter is focused on the overall design and modelling of the Agent-based Controls in the ITS@CU platform. The chapter has two sections:

- The first section covers the detailed design of the three main types of Agents (Control, Operational and Service Agents) and their sub-types/roles.
- The second section describes the overall organisational structure of the Agents and their relationships

4.1. Agents design description

4.1.1. Types of Agents

This research employed a multi-agent approach to control and manage the distributed traffic controllers and systems. Hence the ITS@CU platform comprises of Agents having specific roles and organised into different agencies. These Agents are classified into the following main types:

- **“Control Agent”** for controlling the traffic controllers responsible for managing traffic devices such as traffic lights, vehicle count sensors, grid/vehicle controllers, variable message displays and signs. It is the core type of agent in this agent-based controls approach however the following two agent types are the key enablers in the overall multi-agent platform.
- **“Service Agent”** for accessing/providing external or internal Services such as traffic and weather information services.
- **“Operational Agent”** provides supporting roles in the platform such as security, arbitration and management tasks.

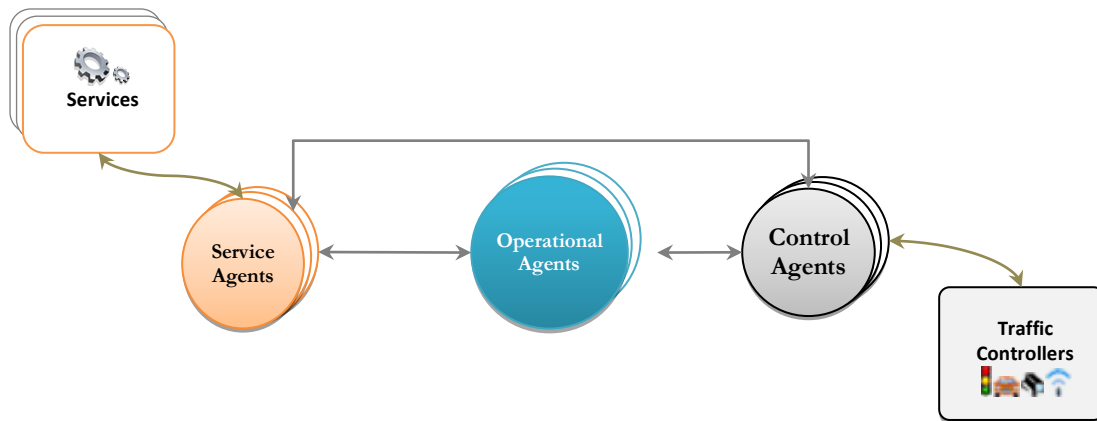


Figure 4.1: Agent types (in ITS@CU platform)

Agent types are further divided into several sub-types based on their role requirements such as capabilities, host/environment, and area of influence. Agents use templates to facilitate the creation of Agents dynamically and provide the flexibility to create customised instances of Agent role/sub-types (Tagni & Jovanovic, 2006). The template provides the means to specify common as well as specific properties and functions. For example, the Control Agent’s sub-types “Traffic Light Control Agent” and “Vehicle Count Sensor Control Agent” have shared common properties and functions (e.g. Location) but have different specific properties and functions (e.g. the

Traffic Light Control Agent has a function to change traffic light/signal duration values which is not present in Sensor Control Agents).

4.1.2. Control Agents

Road network/traffic infrastructures consist of various control systems (roadside sensors, devices, traffic lights, message displays, vehicles etc.), and efficiently controlling and managing these controls are vital in ITS infrastructures. *These controls are divided into the following types.*

- Flow control (Active nodes with traffic lights)
- Informative (Variable Message boards, signs etc.)
- Monitoring (Traffic data collection sensors)

In modern urban traffic management systems, a set or group of specific traffic control devices (e.g. traffic lights or sensors) is controlled by a local “Controller” which is responsible for their operations. The traffic control devices perform specialised tasks and have limited memory and resources. However, Controllers have reasonable CPU, memory, storage and networking capabilities. The Controllers are inter-connected and monitored by central or regional control centres. So if any particular traffic control device stops functioning, the Controller relays a notification message to the control centre system. For example, in the Coventry City traffic management system, all the traffic light and inductive loop detectors/sensors are controlled by their local Controllers (called out-station transmission unit (OTU)) which are interlinked (using built-in modem) and integrated with SCOOT system for overall monitoring and calibration of traffic flow and controllers (see “*Appendix L*” for details).

There are some limitations to this type of approach:

- The controllers have a fixed set of features/functions
- Traffic diversion requires manual intervention
- Traffic light/signal adaptation is very limited
- Communication overheads as each controller is directly communicating with regional/central control system
- No integration with external services at controller level
- Most importantly, it is very costly to update traffic controls/system as update usually means replacing the existing controls.

This research has addressed these limitations by introducing Agent-based controls i.e. Control Agents hosted on Controllers (discussed in *chapter 2, section 2.4*). Agent-based controls extend the concept of agents and fixed controls to a next level by dynamically adding extra functionality and decision making capabilities to the traffic controls. As seen in *figure 4.2*, all geographically distributed local traffic Controllers are governed and controlled by Control Agents which interact with other Agents over a wide area network on behalf of Controllers.

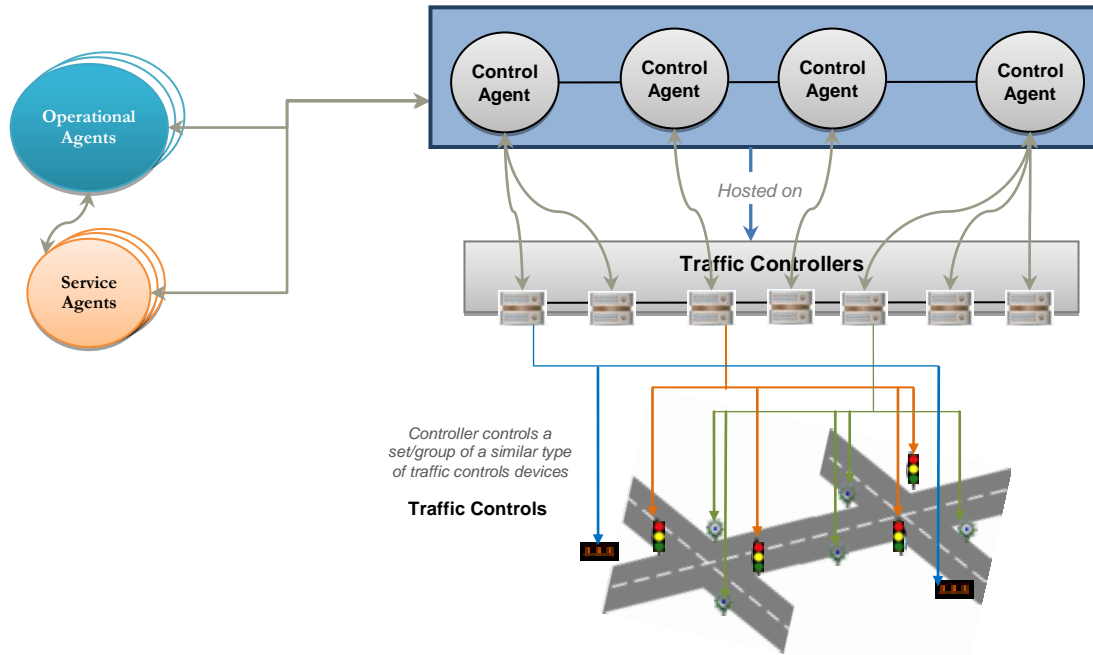


Figure 4.2: Agents-based Controls Overview

A “Control Agent” is hosted on a Controller and it uses the host’s resources such as CPU, memory and network capabilities. Different Control Agents can reside on a Controller at different times however only a single Control Agent will be hosted by a Controller at any given time and will use a pre-configured resource allocation.

In a traditional control system, operational control algorithm/logic is built-in within an isolated Controller and is responsible for all its operations. However, in the Agent-based Control approach, the Controller’s operational logic has been split into “**Default-logic**” and “**Agent-logic**”.

Default-logic is the basic default functionality/logic of a controller, which is pre-configured within the control (e.g. fixed time duration of traffic lights) to enable it to operate despite loss of communication.

Agent-logic is an additional intelligence provided by the Control Agents to the Controllers. It provides dynamic functionalities and decision-making capabilities in response to the current situation of the traffic.

The Control Agent-logic decision making process involves multiple agents working together to assess current situation, arbitration, and data fusion. For example, if a traffic light Controller loses connectivity with the platform, it will stop its dynamic signal adaptation functions provided by Control Agent logic but continue to operate its set/group of traffic lights in a fixed time manner.

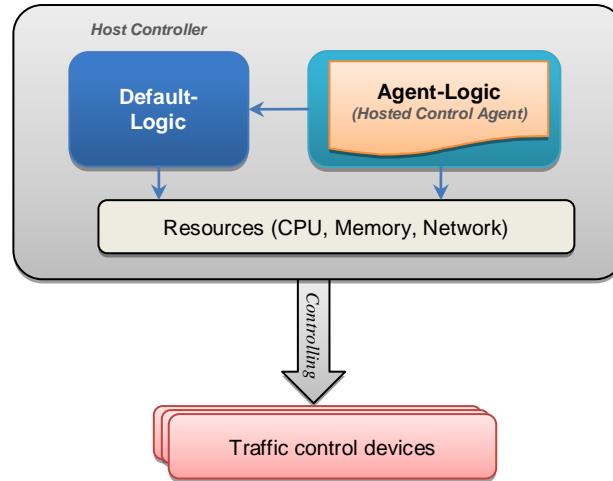


Figure 4.3: Controller with a hosted Control Agent

In the ITS@CU platform only certain devices can become a host for a Control Agent such as traffic light controllers or sensor controllers. Traffic control devices, such as the actual traffic lights, cannot be a host as they are controlled and managed by Controllers, and above and beyond the traffic devices do not have enough processing or memory to host any control agents. As Controllers are responsible for the control devices' functions it is sensible to host Control Agents on Controllers in order to control the networked traffic control devices.

A single Control Agent can be hosted on multiple Controllers which mean a single Control Agent can manage/control multiple Controllers at the same time. This makes this approach very efficient where single Agent deal with multiple Controllers to reduce the communication overheads (between controls, controllers and the central/regional traffic control systems). The Control Agents approach also enables the upgrade or implementation of new traffic management systems/controls without increasing or modifying the capacity of local hardware and software environments. It is a cost effective way to develop, maintain and expand/upgrade a transportation system in connected environments.

For the purpose of resiliency and load balancing a pair or a cluster of Agents can be configured to perform a single agent's task. In such cases one Control Agent works in a primary/active mode and the other(s) in standby/passive mode.

The following types of traffic controllers can be controlled and managed by Control Agents:

Controllers	Notation	Description
Dynamic Traffic Sign	<i>CtrSgnD</i>	Controls a set of connected variable traffic signs (speed, stop etc.)
Variable Message Display Board	<i>CtrMsgD</i>	Controls a set of connected variable Message display boards for road users information
Traffic Lights	<i>CtrTL</i>	Controls a set of inter-linked and dependent traffic lights/signals

Vehicle Count Sensor	<i>CtrlVC</i>	Controls a set of inter-linked vehicle count sensors
Vehicle Controller	<i>CtrlVeh</i>	Each vehicle has a controller which controls its internal controls
Grid Controller	<i>CtrlGrid</i>	Controls a Grid area, as per the road network model
Node Controller	<i>CtrlNd</i>	Controls a Node (intersection/Junction etc.) – <i>Optional</i>
Link Controller	<i>CtrlLnk</i>	Controls a road link – <i>Optional</i>

Table 4.1: Controller types

More controllers can be added however currently only the above are supported as part of the ITS@CU platform.

Roles/Sub-Types of Control Agents

The sub-types of Control Agents designed to manage the traffic Controllers are given in *table 4.2*:

Control Agent Role (Sub-Type)	Controller	Description
<i>cAgtCtrlTL</i>	Traffic lights	Provides Agent-logic and dynamic functionality to the controller of traffic lights/signals to dynamically adapt the signalisation based on multi-agent coordination.
<i>cAgtCtrlVC</i>	Vehicle Count Sensors	Provides Agent-logic functions to the Controller of a set of vehicle count and monitoring sensors for obtaining vehicle flow data in response to emergent situations or request by other agents.
<i>cAgtCtrlMsgD</i>	Variable Message displays	Provides Agent-logic functions to the Controller of Variable Message display boards to intelligently display useful messages based on situations, location and time.
<i>cAgtCtrlSgnD</i>	Dynamic signs	Similar to Message display, it provides Agent-logic functions to the Controller of Dynamic Sign boards to intelligently display different traffic signs based on situations, location and time.
<i>cAgtCtrlGrid</i>	Grid controller	It is the main Agent providing Agent-logic functions to the Grid controller system. It is responsible for overall grid operations and for controlling grid level Operational Agents.
<i>cAgtCtrlCc</i>	Central control System	It is the main Agent controlling the Central Control system. It uses other operational Agents for achieving most of its tasks (mentioned in operational Agents section in the chapter).
<i>cAgtCtrlVeh</i>	Vehicle control	Manages the intra-vehicle functions and interact with other agents either vehicle to vehicle or vehicle to grid controllers
<i>cAgtCtrlLnk</i>	Link	Manages a Link (part of road) in conjunction with above control

		agents on that road. It is though optional agent and can be configured only if required
<i>cAgtCtrlNd</i>	Node	Manages a node (junction/intersection) in conjunction with above control agents on that junction/intersection. It is an optional agent and should only be configured if required

Table 4.2: Control Agent sub-types/roles

Control Agent Design Description

The following is an example of a Control Agent hosted on a Controller (for traffic lights management).

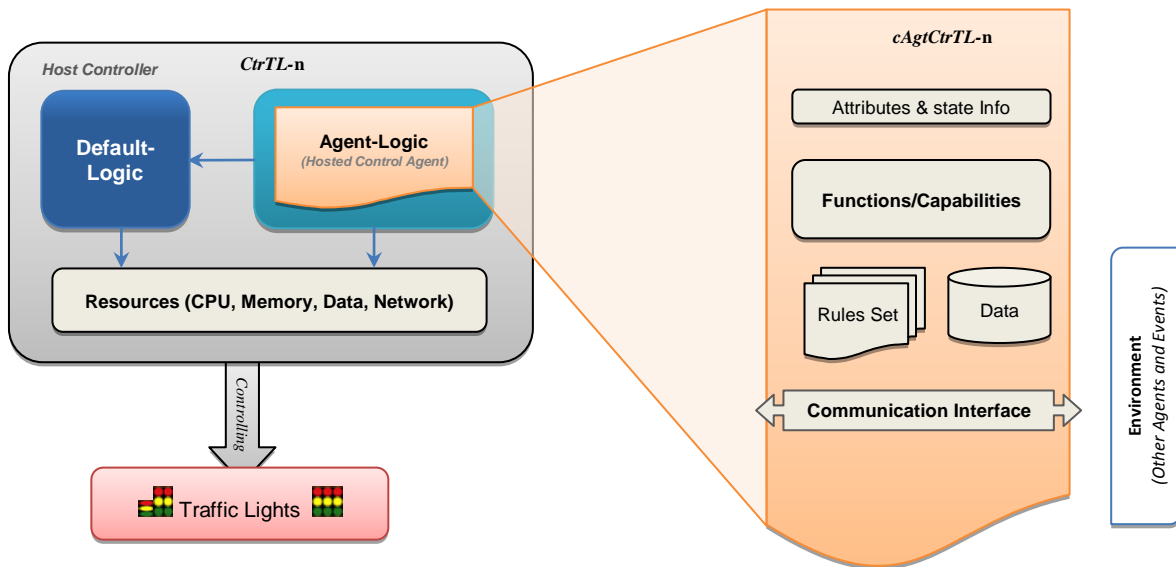


Figure 4.4: Controller and Control Agent (Traffic Lights example)

The Control Agent has the following characteristics and properties:

- A Control Agent is identifiable and has a unique ID which is composed of $cAgt[type]-[n]$ where $cAgt$ represents Control Agent, $type$ is the type of Controller the Agent is hosted on, and n is an incremental and unique value. “ $cAgtCtrl-01$ ” is an example ID of a Control Agent for a traffic lights Controller.
- A Control Agent is an intelligent agent type with hybrid/layered architecture. It is discrete, autonomous and self-contained entity. It has a set of characteristics and logic/rules for governing its internal

behaviours/functions and decision-making capability. An agent exercises control over its actions and states, and it functions independently within its environment and in its dealings with other agents. However, for security and reliability purposes it has a pre-configured role and an area of influence/boundary within which it operates. Also, specialised Operational Agents “Control Agents Manager” can intervene in a Control Agent’s operations, if required.

- A Control Agent is hosted on a Controller and uses the host resources such as CPU, memory, data storage, and networking capabilities. A single agent can be hosted on multiple controllers at the same time forming a “one to many” relationship however a Controller is only allowed to host a single Control Agent at a time forming a “one to one” relationship.
- A Control Agent interacts with the host Controller to command actions or request the use of a resource. It also interacts with other Agents to get information or coordinate in the decision making process.
- A Control Agent is goal directed and performs tasks based on internal or external requests/events. An Agent’s goal can be a subtask delegated by other Agent(s) to achieve a wider/system level goal. For example, multiple Agents can work on traffic signal adaptation for traffic flow optimisation or diversion where each Control Agent performs its internal tasks based on the request by other Agent(s).
- An Agent belongs to an Agency (organisation of agents).
- An Agent can be standalone, part of a team or multiple teams.
- A Control Agent has a defined role and performs a specific set of functions *e.g.* A Traffic Light Control Agent is only capable of managing Traffic Light Controller and a Message Board Display Control Agent is only capable of dealing with Display Controllers. This approach of having specific purpose Agents has advantages in terms of flexibility, reliability, maintainability and update.
- A Control Agent can continuously operate however it can move into other states *e.g.* suspended or idle mode. An Agent can be resumed by either operational agents or any relevant external or internal events (on a host Controller).
- A Control Agent stores its functional capabilities and knowledge (rules/logic, data, ontologies etc.) in the host Controller’s memory/storage. This information can be modified by the agent itself based on situation changes or an update request by other operational Agents. This capability makes Control Agents dynamic, flexible, and having the ability to learn and build a knowledge base by storing rules and past data. The update mechanism also means that an Agent can update its logic/behaviour or even default-logic when new system functionality updates/enhancements are released by Operational Agents.

The Control Agent design structure properties that can also be used as the design template for creating new Control Agent instances is illustrated in *figure 4.5*

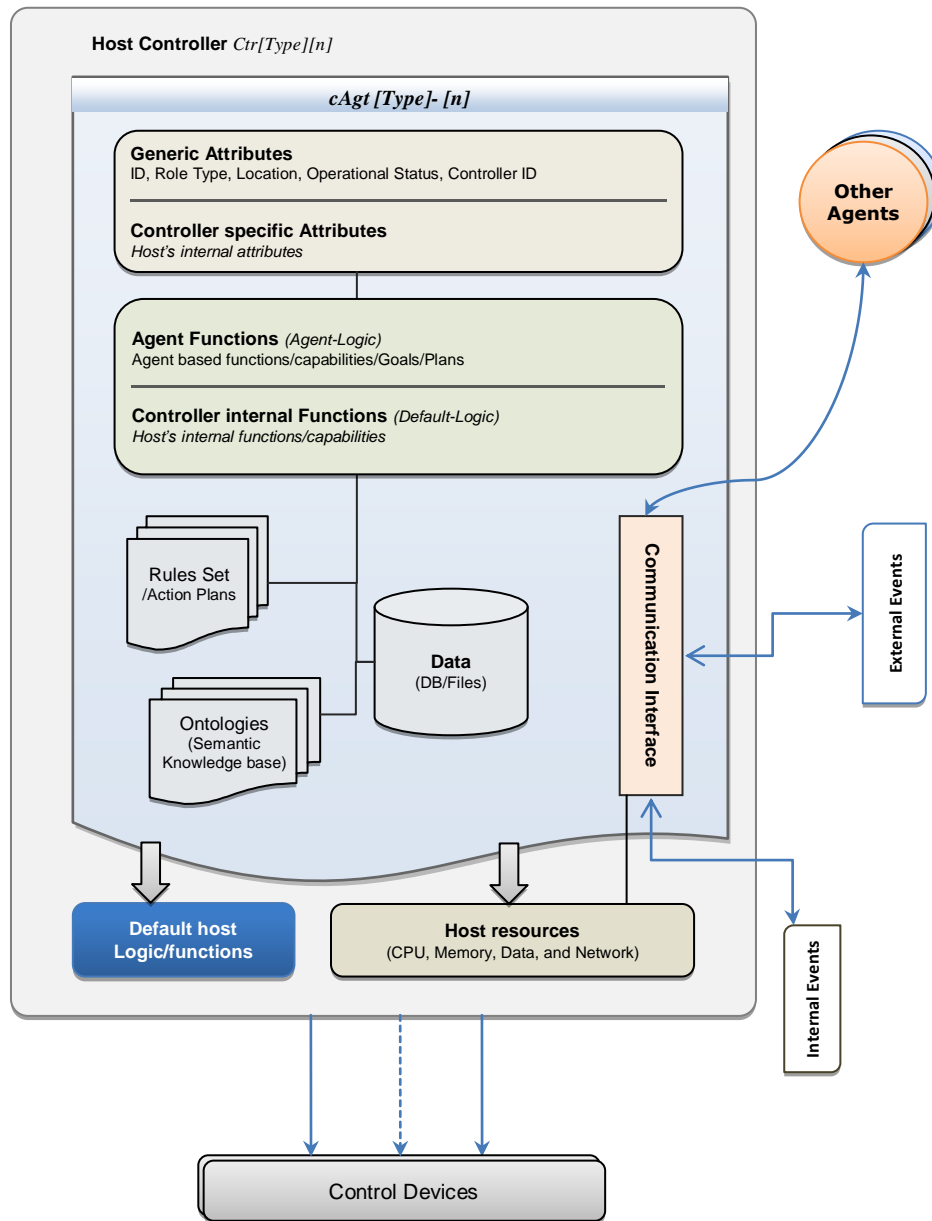


Figure 4.5: Control Agent Design Structure/Template

A Control Agent design shares the same base structure with Operational and Service Agents so they can inherit common communication and knowledge management methods.

Attributes

A Control Agent has several attributes divided into “Generic” and “host specific” types. The following are the base generic attributes:

Attribute	Description
ID	Unique identifier of the Agent
Role-Type	Sub types mentioned in <i>table 4.2</i> , e.g. <i>cAgtCtrl</i> - Traffic Lights Control Agent
Controller ID	Identifier of the host Controller (mentioned in <i>table 4.1</i>)
Location	longitude/latitude of the current host controller(s)
Operational Status	Current status of the Control Agent e.g. active, idle, suspended etc.
Team IDs	The team identifier, if a Control Agent is part of team (or teams)
Agency ID	Identifier value of the organisation the control Agent belongs to

Table 4.3: Control Agent generic attributes

Host specific attributes are different from host to host for example a traffic lights controller has different attributes when compared to a Grid Controller. These attributes are internal to the host and the Agent can use these values for its function and, if allowed, can manipulate these values. Some examples of host specific attributes include Green time of a traffic lights Controller, Vehicle flow detection frequency for a Vehicle Count Sensor Controller, Current hardware status of Controller and traffic controls etc.

Capabilities/Functions

A Control Agent's capabilities are defined as the operational functions or methods which enable it to perform various actions to achieve its goals and plans. These capabilities are either default-logic or Agent-logic based. Default-logic consists of the pre-configured capabilities of the host controller. Agent-Logic consists of the methods/functions which control when, how and with which other agents and hosts the agent communicates with, and affect the behaviour or reasoning of the agent. It also includes functions to use/manipulate attributes and local data using the embedded logic in response to events or interaction with other agents or the host.

Some of the core functions of a Control Agent (Agent-Logic) include:

- Change agent status and Controller's status
- Formulate goals and action plan
- Request information from other Agents (Send messages)
- Raise events or request other agents to show their intentions (ontology requests)
- Update/Modify Agent-logic, rule set and ontology
- Update/Modify default-logic of the Controller – permanently or temporarily

The intentions of an agent are described in terms of imperative goals. Multiple goals can be executed in parallel (provided the host can support multithreading). Goals are either basic actions or sequential compositions of internal functions combined with external Agents functions.

Controller Internal functions (default-logic): Default functions are different depending on the type of host controller and its purpose. Some example functions include:

- Change the controls status and parameter values (e.g. set values of traffic light's green time)
- Sending task execution commands to host (e.g. display traffic message on dynamic display board)

Agent communication interface

The communication of the Control Agent with its environment is an integral part of its operations. It requires external communication with other Control and Operational Agents as well as internal communication with the host Controller (for sending commands to traffic control devices). The external communication interface comprises of light-weight RESTful web services allowing interaction with other Agents and dealing with external requests and events (such as status updates and agents communication requests).

In the event of no connectivity, the host will stop using the Agent's functions (in Agent-Logic mode) which are dependent on other agents' input, and the default-logic mode will take control.

The communication interface has various modules which perform various functions such as sending/receiving messages, queuing, message parsing, ontology interpretation and message composition. Further detail of the communication structure is covered in *chapter 5 and 6*, and the implementation/technological details are covered in *chapter 7*.

Remark: The Controller in default-logic mode still uses the same communication layer and communicates with Operational Agents but without the semantic-content and the intelligence provided by Control Agents. It is also not capable of communicating with other Controllers hence the communication becomes more centralised.

Host resources usage

As seen in *figure 4.5*, a Control Agent hosts itself on a Controller to provide additional logic/intelligence (in addition to Controller's default-logic mode). A Control Agent is hosted by a mutual agreement between the participating host Controller and a "Controller Manager" Operational Agent. An Agent uses the host resources on a pre-defined level as per the agreement between the Controller and "Controller Manager" to avoid misuse and overloading the Controller.

A Control Agent uses host resources to perform the following operations:

- Store its state info in the local data store which can be a database or a file depending on the type of host Controller e.g. Grid controller can host light version of DBMS (SQL Server Compact Edition) however a

vehicle count sensor Controller can only store basic XML files. Similarly, Control Agent stores and accesses its behavioural Rules set and Ontologies (which are defined in XML format) either as files or in Database.

- Communicate with other Agent or local traffic Control Devices using host's internetworking capabilities which may be LAN, wireless or ad-hoc.
- Uses CPU and memory for executing its functions, analysing events and other operations for decision-making process based on the rules, ontologies or other data.
- Exploit the default-mode capabilities of the host for executing simple internal tasks

Local Data, Rules and Ontologies

An agent stores various data locally during its lifetime based on which it performs all its operations and decision making. Different types of data are stored by the Control Agent such as state information, Rules set and Ontologies. All the data are in XML format however the storage type can be a database or a file depending on the host.

Rules form the beliefs of an agent. It is a set of match cases and possible response actions which the agent uses to deal with events and its operations. Ontologies are required for Agents-based controls to understand the semantics of any agent message. Both ontologies and rules are defined in XML format (further detail is covered in *chapter 6*.)

Agent creation

“Control Agents Manager” operational Agent hosted on a grid controller creates a new instance of Control Agent using Agent role-type templates and then assigns a unique ID. Templates are in XML format containing the definition of the classes, behaviour, rules and other design information which are used by the host application to generate multi-threaded objects of the Agent type class (Further explanation is covered in *section 4.1.4* and *chapter 7*).

4.1.3. Service Agents

Services are at the heart of SOA based systems architecture which is also the core architecture of the ITS@CU platform. There are various types of services in the platform providing various features and functions ranging from internal communication layer services to external sources e.g. weather services and traffic information service.

In the ITS@CU platform, all external Services are governed by dedicated Service Agents which serve as brokers between a Service and consumer Agents i.e. Control, Operational and Service Agents. For example, to access the weather information service from an external source (such as the Met Office), the ITS@CU platform has a dedicated Service Agent which all other Agents use to access the service. The dedicated Service Agent holds all the details such as Service description (WSDL), authentication rights and other service agreements with the Service.

Keeping Operational and Control Agents separate from direct interaction with the services makes the platform's approach very flexible and also avoids unnecessary communication. For example, if Control Agents or Operational agents interact directly services then they have to know all the available services and their features in the platform which means they require all the services description/WSDL, Authentication details, service agreements and other details, resulting in an additional and unnecessary processing overhead for normal Agents. Another benefit of the Service Agents is if any Service is updated only the relevant Service Agent requires updating and no other component of the platform is affected. This approach provides flexibility, and enables an efficient and cost effective way to maintain and upgrade the ITS@CU platform's components/agents.

Agents internally communicate with each other using a described communication structure however dedicated Service Agents can communicate with external services based on the service's source implementation technology e.g. SOAP, RESTful, WCF or any SOA enabled service platform. Moreover, Service Agents have the capability to encapsulate semantic data in the form of domain ontologies in order to understand and communicate with semantic web services.

Service Agents can interact with a single atomic service or can combine different services into a composite workflow, for example a single service agent can communicate with different weather traffic information services at the same time and combine the result back to the requester/consumer Agent(s).

Service Agent Design Description

Service Agents by design are reactive type of agents as they only respond to requests made by other agents, and do not require proactive capabilities. As seen in *figure 4.6*, all external and internal Services are accessed and provided by Service Agents. Other types of Agents interact with Service Agents over a wide area network.

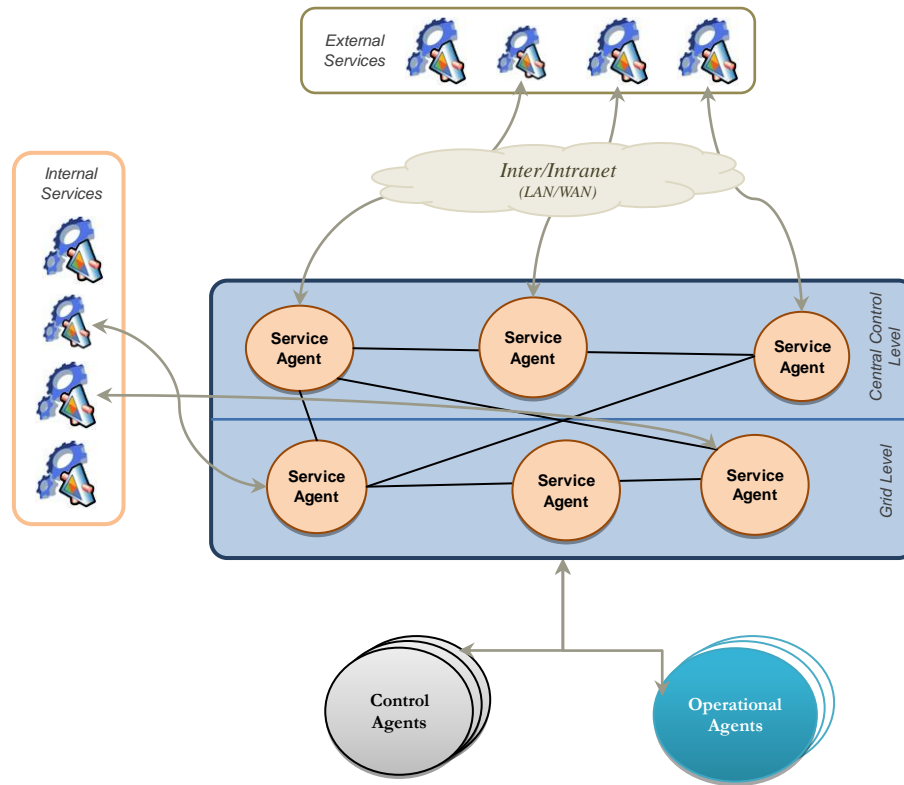


Figure 4.6: Service Agents Overview

The ITS@CU platform can support any type of Service provided the service conforms to SOA principles and W3C standards. The Services implemented are:

Services	Notation	Description
Weather Information	<i>SvcWS</i>	<p>A service providing real-time weather information based on street level location. Various service providers are currently available such as the Met Office, Yahoo UK and BBC allowing web service integration.</p> <p>A web service application was developed for simulations which provides similar functionality but with pre-configured data which can be altered in real-time to simulate different weather conditions.</p>
Route Information	<i>SvcRt</i>	A simulation purpose Web Service application was developed for the platform evaluation. It provides simulation route information data which can be configured to street level.
Bing Map	<i>SvcBing</i>	<p>It provides two online features:</p> <ul style="list-style-type: none"> Reverse geocoding service provided by Microsoft Bing Maps Service.

		<p>It provided the exact coordinates of a location or address and vice versa.</p> <ul style="list-style-type: none"> Route information and direction service provided by Microsoft Bing Maps Service. It is used in diversion planning by Agents.
--	--	--

Table 4.4: Platform external Services

More services can be added however only the above were included as part of the platform at this time.

Roles/Sub-Types of Service Agents

The following sub-types of Service Agents are designed to provide the services, mentioned in *table 4.4*:

Service Agents Notation	Service	Description
<i>sAgtSvcWs</i>	Weather Info	Accesses the Weather Information Service in response to other Agents requests
<i>sAgtSvcRt</i>	Route Info	Accesses Route Information Service and interacts with Agents in the platform (in response to other Agents requests)
<i>sAgtSvcBing</i>	Bing Service	Accesses the online Bing Map Service in response to other Agents requests

Table 4.5: Service Agents Sub-types/Roles

A Service Agent in ITS@CU has the following characteristics and properties:

- A Service Agent is identifiable and has a unique ID which is composed of *sAgt[Service type]-[n]* where *sAgt* represents Service Agent, *type* is the Service type, and *n* is an incremental value. “*sAgtSvcWs-001*” is an example ID of a Service Agent for a Weather Information Service Agent.
- A Service Agent is reactive and performs tasks based on external requests/events. Agent’s goal can be a subtask delegated by other Agent(s) achieving a wider system level goal.
- A Service Agent exercises control over its actions and states, and it functions independently in its environment and in its dealings with other agents. However, for security and reliability purposes it has its pre-configured role and an area of influence/boundary within which it operates. Specialised Operational Agents “Service Agents Manager” can intervene in Service Agents operations, if required.

- A Service Agent has a defined role to provide a single service *e.g.* the Weather Information Service Agent is only capable of interacting with Weather Info Service. This approach of having specific purpose Service Agents has advantages in terms of flexibility, reliability, maintainability and system/Controller features update.
- A Service Agent is usually continuous i.e. constantly running in listening mode however it can go into other states such as suspended and idle.

The following *figure 4.7* illustrates the Service Agent design structure based on the above properties. This structure is also used as the design template for creating new Service Agent instances.

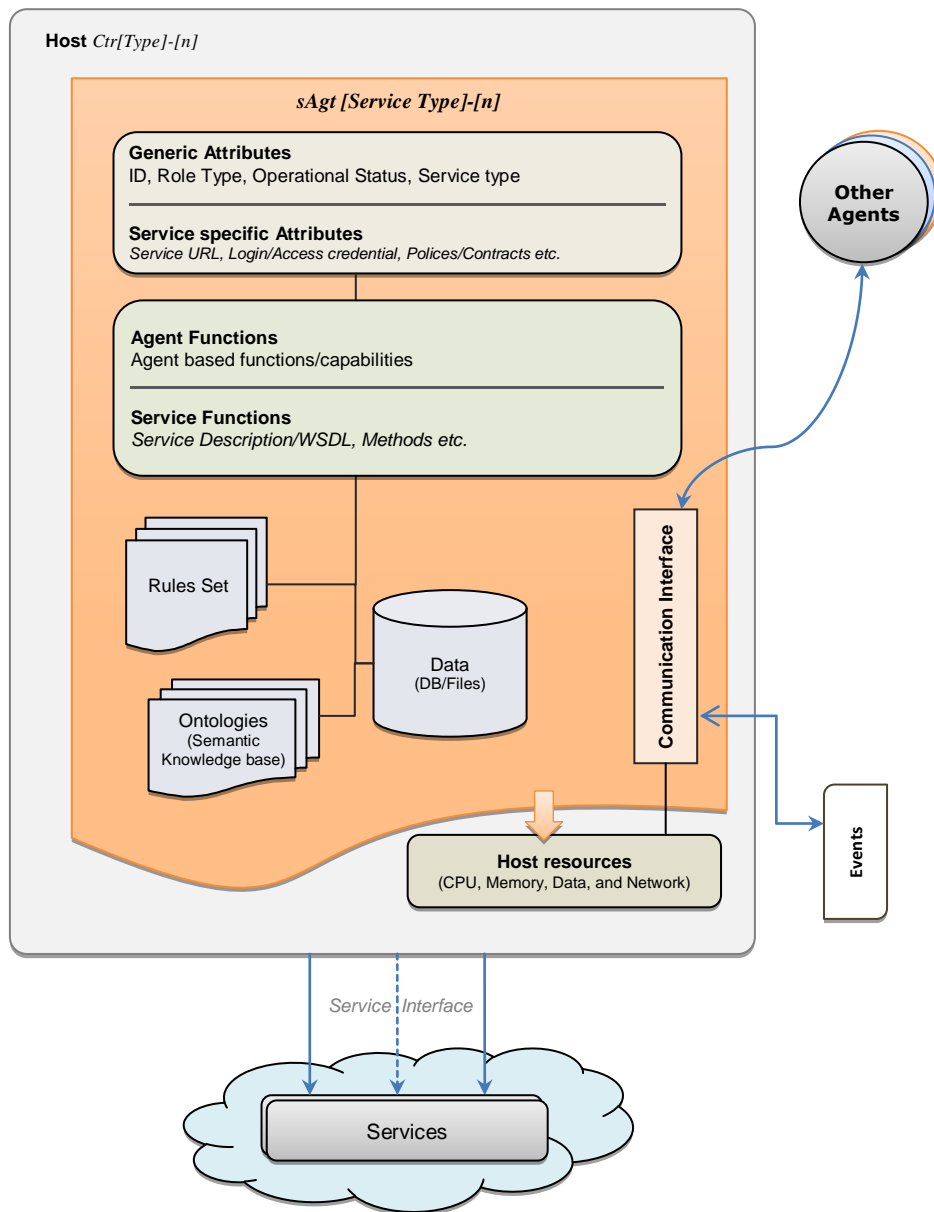


Figure 4.7: Service Agent Design Structure/Template

As seen in *figure 4.7*, a Service Agent hosts itself on a Controller (Central Control or Grid Controller). The Service Agents are hosted on a fixed host i.e. they do not move between hosts.

Service Agents share the same base structure with Operational and Control Agents so they can inherit common communication and knowledge management methods.

Attributes

A Service Agent has several attributes divided into “Generic” and “Service specific” types. The following are the base generic attributes:

Attribute	Description
ID	Unique identifier of the Agent
Role-Type	Sub types shown in <i>table 4.5</i> , e.g. <i>sAgtSvcWs</i> – Weather Service Agent
Host ID	Identifier of the host Controller
Operational Status	Current status of the Service Agent e.g. active, idle, suspended etc.

Table 4.6: Service Agent generic attributes

Additional roles/Sub-Types can be dynamically added, if required.

Service specific attributes are different for different service for example the weather service has different attributes to the Route Service. Some of the examples of service specific attributes include URL of the Service, credentials Login/access info, allowed usage Policies and service Contract info.

Capabilities/Functions

A Service Agent comprises of various functions mainly to interact with the corresponding Service (or other Service Agents) on behalf of the requesting Control or Operational Agents. Some of the core functions of a Service Agent include:

- Change its states and local parameter values
- Update service description/WSDL, internal logic and ontology – *permanently or temporarily*

Service functions are the methods the corresponding Service provides as per its description (such as WSDL). They are required for the service Agent to fulfil the requests by other Agents.

Agent communication interface

A Service agent mainly responds to the requests by other agents and then interacts with Services or other Service Agents. Similar to Control Agent design, the Service Agent itself provides a light-weight RESTful web services based communication interface for external Agents to call/communicate with it. A Communication interface has various modules performing functions such as sending/receiving messages, queuing, message parsing, ontology interpretation and message composition. Further detail of the communication structure is covered in *chapter 5 and 6*, and the implementation/technological details are covered in *chapter 7*.

Host resources usage

Similar to Control Agent design, a Service Agent also requires a host (and its resources) to function. The host must have connectivity (intranet or internet) in order to access the corresponding Service and may also require high bandwidth as compared to Control Agents. A Service Agent requests resources from its host, and interacts with other Service Agents to get information or coordinate for decision making as part of composite service work flows. Usually regional Control Room centres or Grid Controllers are suitable for hosting Service Agents. Unlike Control Agents, multiple Service Agents can reside on a host at the same time as their hosts do not have the resource limitations of traffic Controllers.

In ITS@CU Service Agents that are responsible for external Service access are hosted on the Central Control systems and the Service Agents that are responsible for internal services are hosted at both Central/Regional or Grid Controllers level (details are covered in *chapter 7*).

A Service Agent uses the host's resources (CPU, memory, storage, DBMS and network capabilities) on an allocated level to avoid misuse and overloading the host.

Service discovery and subscription model

As seen in the *figure 4.8*, Agents use the "Service Registry Operational Agent" to find/discover Service Agents. Service Registry Agents keep a registry of all the Service Agents and corresponding Services. A Service Agent can publish/register its capabilities to which other Agents can subscribe, and whenever new changes occur the Service Agent can inform the subscribers.

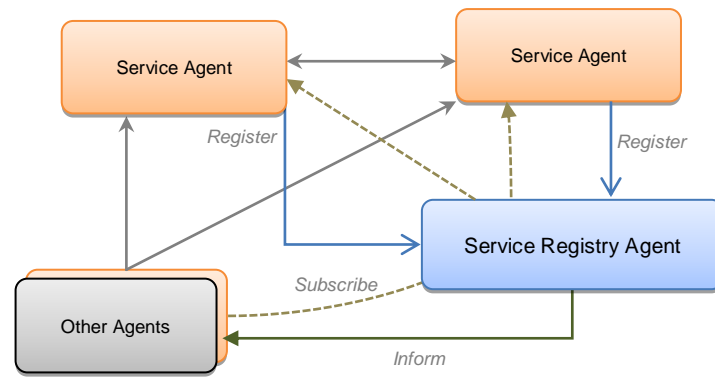


Figure 4.8: Service registration and subscription overview

4.1.4. Operational Agents

Operational Agents provide specialised and supporting roles for Agent-based Controls in the ITS@CU platform such as security, arbitration, facilitation, administration and management tasks. There are different types of Operational Agents with different roles, responsibilities and areas of influence/levels of access. Depending on their role type, they communicate and coordinate with other relevant Operational Agents, Service Agents and Control Agents. They also communicate with their host system components as seen in *figure 4.9*.

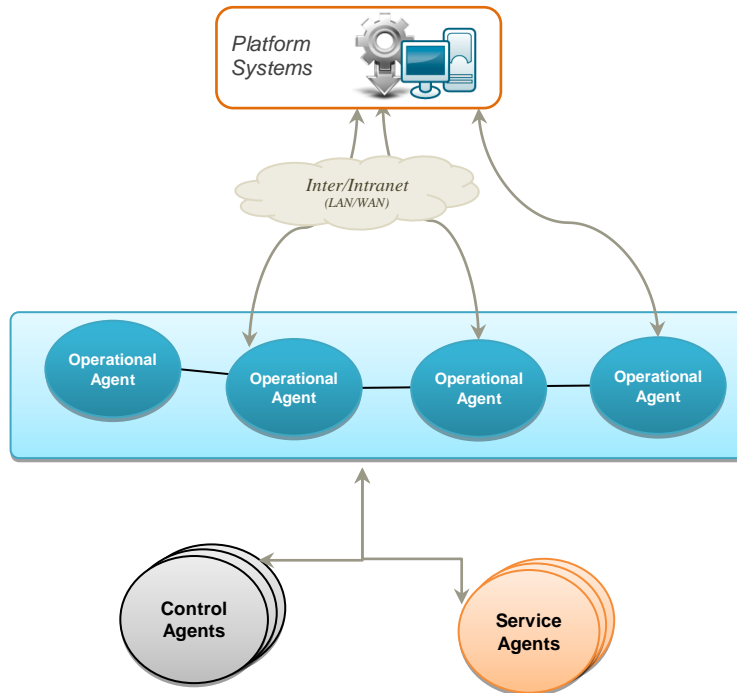


Figure 4.9: Operational Agents Overview

Roles/Sub-Types of Operational Agents

The following types of Operational Agents with distinct roles are used for various purpose in the ITS@CU platform:

Notation	Agent Role	Purpose /Role Description
<i>oAgtArb</i>	Arbitrator Agent (Broker)	Provides Arbitration between Agents. It can be configured to operate at different levels i.e. have access level or area of influence at Grid, Controllers

		or Service level. It is the first point of contact for Agents in different grids or levels, for example, if a Grid Control Agent wants to instigate a communication with other Grid Control Agent, it invokes communication with the Arbitrator Agent of that Grid.
<i>oAgtSec</i>	Security	Authenticates and verifies Agents validity and its level of access
<i>oAgtCtrMgr</i>	Host Controllers Manager	It manages the lists of Controllers (hosts) and their capabilities in the platform. It also facilitates the allocation of Control Agent to a Controller and agrees the resource usage of the host Controller.
<i>oAgtcAgtMgr</i>	Control Agents Manager	Manage Control Agents in the platform: <ul style="list-style-type: none"> • Create a new instance of Control Agent using the design templates • Add, delete or suspend a Control Agent • Keeps track of control agents lists and current host location to facilitate inter-agent communication.
<i>oAgtOAggtMgr</i>	Operational Agents Manager	Manage (add/delete/suspend etc.) and keeps track of all Operational Agents.
<i>oAgtSAggtMgr</i>	Service Agents Manager	Manage (add/delete/suspend etc.) and keeps track of all Service Agents.
<i>oAgtSvcReg</i>	Service Registry	Keep the registry of all the services and the Service Agent currently providing interface to that service. This operational Agent assists in the service discovery (using meta-tags) and dynamic services composition over the service bus of the platform (mentioned in the SOA reference architecture in <i>chapter 7</i>)
<i>oAgtQoS</i>	Agents Quality of Service (QoS)	<p>This Agent provides the current performance and operational status of the platform's agents in terms of their Quality of Service (QoS) values. The agents QoS parameter values changes based on their current performance i.e. response time (which are configurable values set in milliseconds <i>ms</i>). The following QoS values are currently setup:</p> <p>3 = Normal (Response time less than the normal <i>ms</i> value) 2 = Overloaded (Response time above the normal <i>ms</i> value) 1 = Unresponsive (not responding within the maximum <i>ms</i> value) 0 = Stopped</p> <p>This agent periodically checks the status of all active Agents in the platform by sending ping broadcast messages (interval configured as every 3 minutes), and then it assigns a QoS value to each agent based on the response time.</p> <p>Other agents use these QoS values to choose an Agent to carry out task(s),</p>

		<p>where multiple agents with similar capabilities for the required tasks are available.</p> <p>In the case where an Agent has 0 or 1 value, it informs the relevant Manager Agent (e.g. <i>oAgtAgtMgr</i> for Control Agents) to take an appropriate action (e.g. create new instance) based on the situation as programmed.</p>
<i>oAgtPoolMgr</i>	Pool Manager Agent	In cases where multiple Agents work as a pool or cluster for load balancing/sharing, one of the Agents in that pool also takes the role of Pool Manager Agent. It then deals with all the external requests and decides which agent within the pool can be used by taking into account the QoS values provided by the <i>oAgtQoS</i> agent.
<i>oAgtSys</i>	System Agent	General purpose agents for system management operations such as Recovery, clean-up etc.
<i>oAgtAppUI</i>	Interface Agent	Application simulation purpose only - <i>Optional</i>

Table 4.7: Operational Agents Sub-types/Roles

Operational Agent Design Description

An Operational agent is both pro-active and reactive depending on its role. It performs tasks based on its own or on external requests/events. Some Operational agents also have mobile capabilities and can migrate between different hosts to perform its goals. The goal of an agent can be a subtask delegated by other Agent(s) achieving a wider system level goal.

The Operational Agent in ITS@CU has the following characteristics and properties:

- An Operational Agent is identifiable and has a unique ID which is composed of *oAgt[type][n]* where *oAgt* represents Operational Agent, *type* is the Agent sub-type, and *n* is an incremental value. “*oAgtArb-01*” is an example ID of Arbitrator Operational Agent.
- An Operational Agent is an autonomous entity with a set of characteristics and logic/rules for governing its behaviours/functions and decision-making capability. An agent exercises control over its actions and states, and it functions independently in its environment and in its dealings with other agents. However, for security and reliability purposes it has its pre-configured role and an area of influence/boundary within which it operates. Also “Operational Agents Manager” can intervene in the Operational Agent’s operations, if required.

- An Operational Agent belongs to an Agency (Agent Organisation), and can be standalone, part of a team or multiple teams.
- An Operational Agent runs continuously however it can go into other states such as suspended and idle. It can then resume its role following a relevant operational agents external requests or internal events
- An Operational Agent resides on a capable host and uses the host's resources such as CPU, memory, data storage, and networking capabilities. An Operational Agent stores its functional capabilities and knowledge (rules/logic, data, ontologies etc.) in the host Controller's memory/storage. All such information can be modified by the agent itself based on situation changes or an update request by other operational Agents. This capability makes Operational Agents dynamic, flexible, and the ability to build a knowledge base by storing rules and past data as experience.

The following *figure 4.10* illustrates the Operational Agent design structure based on the above properties. This structure is also used as the design template for creating new Agent instances.

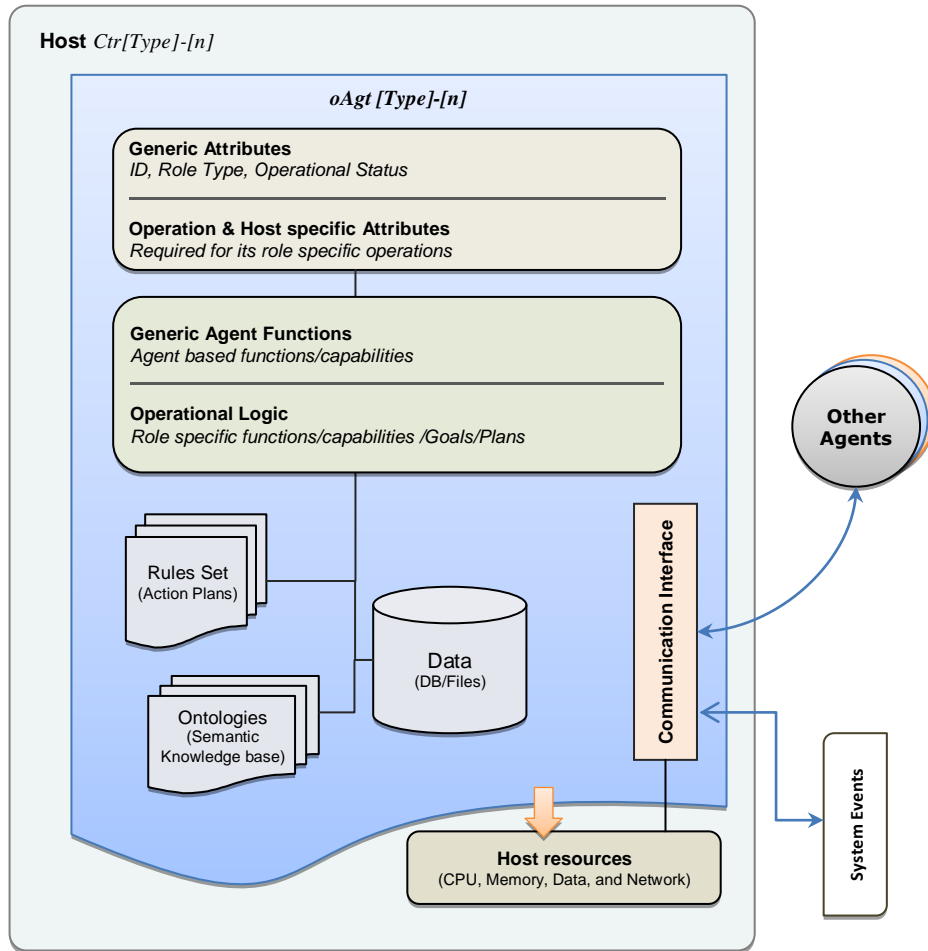


Figure 4.10: Operational Agent Design Structure/Template

Attributes

An Operational Agent has several attributes divided into “Generic” and “Operation & host specific”. Generic attributes are similar to Control and Service agents i.e. ID, Role/Sub-Type, Operational Status, Host ID etc.

Operation & host specific attributes are different for different role types of the Operation Agent and the type of host. These attributes can be internal to its host and the Agent can use these values for its function and if allowed can manipulate these values.

Capabilities/Functions

An Operational Agent comprises operational functions to deal with Agent requests and interact with other Agents and platform components.

Some of the core generic functions include:

- Change agent status and Controllers status
- Request information from other Agents (Send messages)
- Raise events or request other agents to show its intentions (ontology requests)
- Update/Modify Agent-logic, rule set and ontology

Operational Logic is the core of the operational Agent required to perform its functions and execute plans. They are different based on the type of agent’s role and host. For example, Security Agent has functions to check the Access level or credentials of any Agent and QoS Agent have functions to assess Agents health and performance (*mentioned in table 4.7*).

Agent communication interface

The communication interface of operational Agent is similar to Control Agent’s design. It interacts with its host to request the use of its resources and interacts with other Agents using domain specific ontologies. It provides a light-weight RESTful web services based communication interface for external Agents to communicate.

Communication interface modules perform functions such as sending/receiving messages, queuing, message parsing, ontology interpretation and message composition. Further detail of the communication structure is covered in *chapter 5 and 6*, and the implementation/technological details are covered in *chapter 7*.

Resiliency and load balancing

An Agent can deal with multiple requests at the same time however a pool (pair or a cluster) of operational Agents can be configured to perform single agent's task for the purpose of resiliency and load balancing. In such cases one Agent within the pool becomes a pool manager to distribute requests evenly.

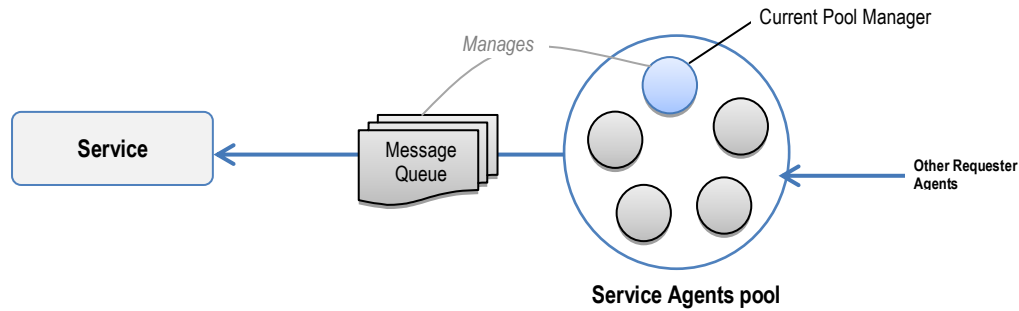


Figure 4.11: Agent pooling and queues

Split-site hosting mechanism

Similar to other Agents in the platform, Operational Agents also require a host (and its resources) for executing its functions, store/access local data and internetworking/communication. Usually regional Control Room centres or Grid Controllers are suitable for hosting Operational Agents. Unlike Control Agents, multiple Operational Agents can reside on a host at the same time as their hosts do not have the resource limitations of traffic Controllers.

Some types of Operational Agents (“Arbitrator” and “System Agents”) are mobile agents i.e. they can move between different hosts dynamically and a single Agent instance can even reside on multiple hosts (split-site hosts) by dividing itself in partial instances (see *figure 4.12*).

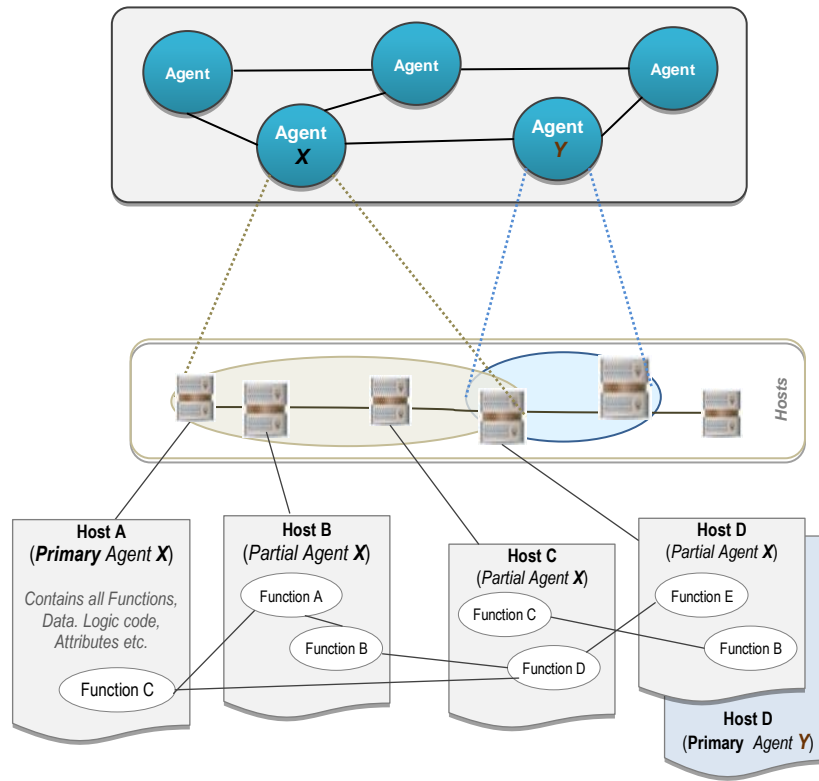


Figure 4.12: Split-site agent hosting mechanism

The split-site hosting mechanism allows a single agent to split its tasks and functionality (Agent-logic) based on host specific requirements or other situations/reasons:

- A host of certain type may only support certain functions due to hardware/software limitation, for example, a traffic controller may have hardware with which requires only specific or compatible functions. As seen in *figure 4.12*, Agent X has been split across four hosts and “Function A” is only hosted on host B as this function is only suitable or used by host B.
- Avoid hosting functions which are not used by the host e.g. a host without Database system does not require functions dealing with databases (store/retrieve queries)
- Require multiple processing at the same time (simultaneous tasks execution)
- The current or next state of the host may require different functions

In this approach of hosting mechanism, there is always one “primary” host and one or more partial hosts. The primary host is the main executing point of an Agent’s logic, and it can move functions between partial hosts dynamically based on requirements and changes on the host.

The primary host is selected based on factors such as resource capacity and proximity/distance. The primary host is not fixed i.e. it can move to other hosts as well. When an agent splits it also checks if there is a better host (using *QoS* Agent), and if necessary it then moves (or re-gathers on a new host).

Using the split-site hosting mechanism in Agents has the following main benefits:

- A single agent can manage operations on multiple hosts systems
- Share load between multiple hosts i.e. avoid overloading resources on a host
- Achieve multi-threading and synchronous task and sub-tasks execution
- Splits its functionality (operational logic code) based on the host's requirement, type and states
- Efficient way of providing localised functions to hosts of different types by the same Agent
- Avoid the use of fat agents where all the code logic is carried by an agent during its movement from one host to another

4.2. Agents organisational structure

Agents form a goal-driven multi-agent society in the ITS@CU platform comprising of different types of Agents with specific roles, belonging to team(s) and Agencies.

An Agency in this context refers to a group of Agents organised hierarchically at the system level. Every Agent must be affiliated with an Agency (and have an attribute Agency ID). The platform has a main Agency called “System Agency” which is at the core platform level. Most Operational and Service Agents are affiliated with the System Agency. Further Agencies called “Grid Agency” are formed at grid/zone level. Control Agents are always affiliated with Grid Agencies, for example Control Agents managing the traffic Controllers in Coventry City Centre can be in one Agency and controls in outer zones can be in other Agencies.

Separating Agents into different Agencies provides flexibility and allows a federated Multi-Agent System which improves maintainability and communication. They are also important for Agent-based controls to work in synergy with other Control Agents within the same Agency. Control Agents from different Agencies cannot communicate directly however if required, their communication can be facilitated by the “Arbitrator” Operational Agents.

An agent can be standalone, part of a team or multiple teams in the approach employed in the ITS@CU platform. Teams refer to Agents working together on a specific task to achieve a common goal. Teams can be formed between Agents from the System Agency and Grid Agencies. No team formation is supported between Control Agents from different Grid Agencies without the involvement of “Arbitrator” Operational Agents at the System Agency level. Teams are dynamically formed and last until the completion of its task/objective. Teams may delegate tasks to their sub teams which can play different roles in different teams.

The following *figure 4.13* shows a team of Operational Agents and Service Agents in the System Agency, collaborating with Control Agents controlling a set of traffic device control (via Controllers).

This research employed multi-agent approach to control and manage the distributed traffic controllers and systems. The ITS@CU platform hence comprises of Agents having specific roles and organised in different agencies

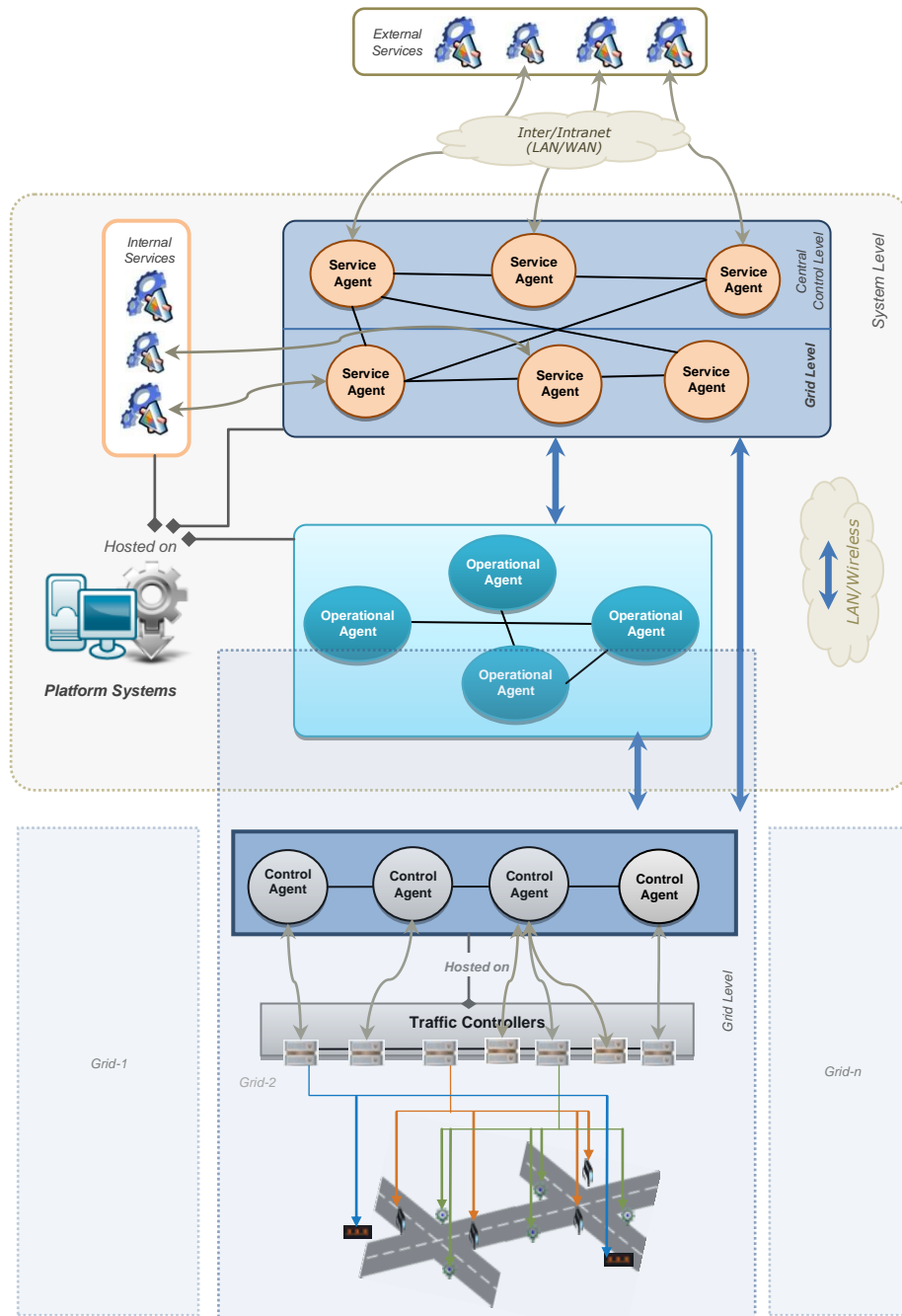


Figure 4.13: ITS@CU Agent Organisational Structure

See “*Appendix J*” for organisational relationship model in form of ontology described in XML Schema

4.3. Conclusion

This chapter described the Agent-based Controls approach which addresses the limitations of fixed traffic control/devices by extending their capabilities to include Agent-logic provided by dedicated Control Agents. Control Agents are supported by Operational Agents and Services Agents in a unique organisational structure where all the agents having specific roles work in synergy to accomplish complex tasks.

Throughout the chapter the reasoning and novelty of the agent design approach was discussed with emphasis on the Control Agent and the concept of Agent-based Control.

Chapter 5

Agent Communication Structure

The previous chapter described the overall design and modelling of the Agents in the ITS@CU platform. It covered the detailed design of the three main Agent types, their sub-types/roles, capabilities and the overall organisational structure.

This chapter is focused on the communication layer and the message instructions used by the Agents (*described in the previous chapter*) to efficiently communicate and coordinate with each other. It is divided into the following sections:

- Overview of Agents Communication Interface/Layer
- Structure details of the Agent Message and Instructions/Commands types
- Agent communication life cycle overview including an example
- Security considerations for Agents communication in the platform

5.1. Multi-agent communication layer

The communication between Agents and other system components (hosts and Services) is a fundamental element of the implementation approach in the ITS@CU platform. It is a novel agent-based communication mechanism specifically designed to support Semantic Agent-based Controls.

The communication between Agents takes place in form of messages with a set of predefined instructions understood by all Agents and specific message semantic content (domain and Agent type specific). The agents possess both individual and collective level decision making capabilities using dynamic rule sets and plans. The messages instructions and semantic contents allow agents to coordinate, collaborate and negotiate amongst themselves.

As seen in *figure 5.1*, the platform uses the “Agent Communication Layer” which comprises of various components to handle the messages between agents. It manages incoming messages/events from other agents or outgoing messages intended for other agents. It is also designed to support semantic web service approach allowing the augmentation of semantic metadata within Agent messages.

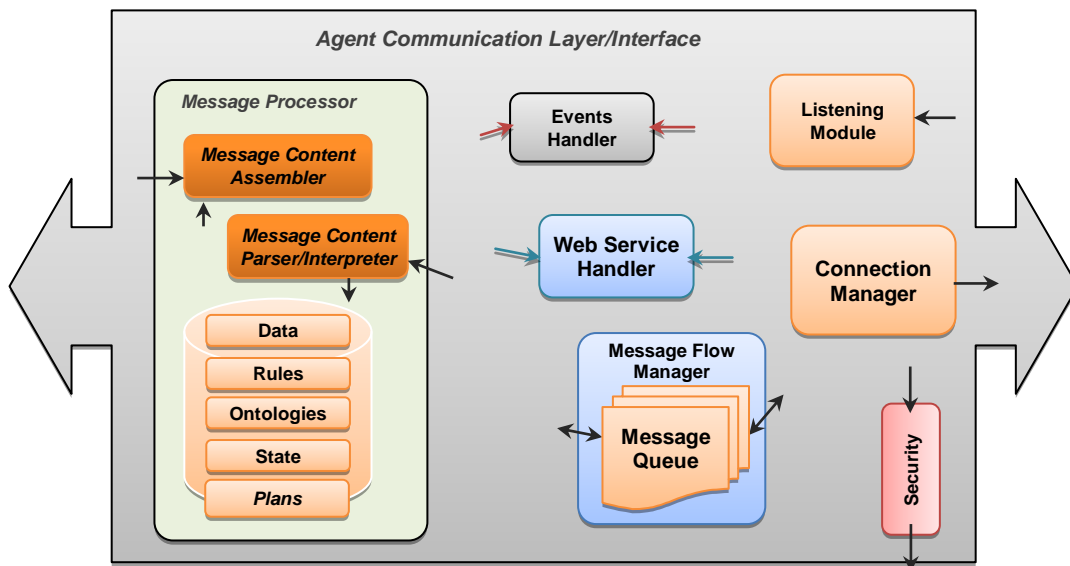


Figure 5.1: Agent Communication Layer

The “listening module” actively looks for incoming messages either from Agents request or events from the system components or host controllers. All messages (incoming, outgoing messages and events) are managed by the “message flow manager” which puts each message in a queue based on its priority. The message flow manager also manages the sequence of function execution (as part of the message flow) especially in the split-site hosting mechanism of mobile operational Agents (Arbitrator and System Agents). This sequencing is very

important for messages that are part of a workflow for example if the message is a request which is part of the subtask delegated by other agents and the request either has to happen at a specific time or event or requires further input from the environment (other agents or host).

The “events handler” deals with internal or external events based on the local rules or data.

The “web service Handler” is a special module capable of dealing with web service based requests. This module can interpret different types of web service-based technologies such as REST, SOAP or WCF. This is important for SOA enabled messaging.

Once a message is released by the message flow manager, the “message content parser” module interprets the actual content of the message, which is encoded in domain specific semantics (ontologies defined in XML format). The message contents once interpreted against the local ontology and rules are then parsed for further processing.

Similarly, if an agent wants to communicate with other agents it assembles the message and dynamically creates the content of the message based on its intention or plan. The newly created message is handled by the message flow manager which puts the message into the message queue (based on its priority or sequence). The message intended for another agent is then dealt with by the “connection manager” module.

The “connection manager” is responsible for establishing a connection with the target recipient agent (using relevant discovery operational agents). The connection manager uses its credentials in order to connect with other agents. It also manages the sessions and other connectivity related functions (message queuing and connection re-establishment in dealing with connection lost/issues).

All messages or events received by an Agent are validated against the “security module” which communicates with a security operational agent to perform security checks.

5.2. Agent message structure

As seen in *figure 5.2*, the structure of the agent-based message comprises of:

- **Message instruction:** Outer message layer with instructions (communicative acts) understood by all agents in the platform. They contain information such as sender and recipient IDs, type of message, type of message instructions, timestamp of creation, priority etc.
- **Semantic-content:** This is a semantic layer encoded within the agent message. The semantic-content contains domain specific commands representing the actual intention of the sender Agent. These semantic commands are different for different types of agents and are only understood by the recipient agents with the domain knowledge (ontology) and the agent-logic to deal with the message content. The semantic layer is highly flexible and the semantic-content commands can be anything from a simple information request to a very complex set of actions/sequence of actions.

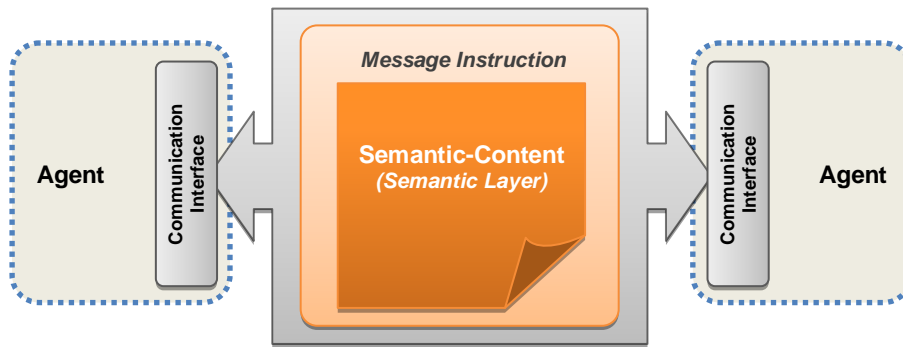


Figure 5.2: Agent message communication structure

Every agent message is unique and identified by a message ID to avoid any ambiguity. The following *code excerpts 5.1* and *5.2* show an agent message example with message instruction and semantic-content. It is a simple message exchange between a Traffic Light Control Agent requesting the current occupancy of a road from the relevant Link Control Agent.


```

<?xml version="1.0" encoding="utf-8"?>
<Agent-Message Message-Type="Request" Instruction-Type="Action" Priority="high"

    Message-ID="MSG00000123" Conversation-ID="CON-MSG00000123"
    Timestamp="2010-11-11 19:23:44" Expires="2010-11-11 19:25:44">

  <Sender-List>
    <Agent ID="cAgtCtrTL-01" IsOriginator="True" IsCurrent-Owner="True" />
    <!-- Other optional Agent details e.g. Agency, type etc. can be listed here, however, not mentioned in this example -->
  </Sender-List>
  <!-- List of other Senders, if the message was part of chain/workflow. However in this example the Agent is both
    Sender and Originator, so no other senders are listed -->

  <Recipient-List>
    <Agent ID="cAgtCtrLnk-07" IsPrimary="True" />
    <!-- Can have other elements, attributes such as Info Only, Dependency, Sequence ID etc. -->
  </Recipient-List>
  <!-- List of other Recipients, if the message was intended for multiple recipients as part of message chain/workflow -->

  <Semantic-Content >
    <Ontologies-List>
      <Ontology Priority-Order="1">Traffic-Flow</Ontology>
      <!-- Can be multiple Ontologies and more attributes such as context, location, version etc. -->
    </Ontologies-List>
    <Context-Expression-Value>
      <!-- The rest is only understood by the relevant Recipient -->
      <Get-Link-Occupancy filter="">
        <Link-Name>Foleshill Road</Link-Name>
        <Start-Node>CtrNd-005</Start-Node>
        <End-Node>CtrNd-006</End-Node>
        <!-- Can have various other elements/Attribute based on ontology -->
      </Get-Link-Occupancy>
      <!-- Can have various other elements/Attributes -->
    </Context-Expression-Value>
    <Rules /> <!--Rules relevant to the message, if required -->

  </Semantic-Content>

  <Conditions /> <!--optional, statement on which the request has to be fulfilled -->

</Agent-Message>

```

Message

Semantic Content
Only understood by the relevant Recipient

Code excerpt 5.1: Request Message Example

```

<?xml version="1.0" encoding="utf-8"?>
<Agent-Message Message-Type="Reply" Instruction-Type="Response-Result"
    Message-ID="MSG00000124" Conversation-ID="CON-MSG00000123"
    Timestamp="2010-11-11 19:24:01" Expires="2010-11-11 19:24:01" >

  <Sender-List>
    <Agent ID="cAgtCtrLnk-07" IsOriginator="False" IsCurrent-Owner="True" />
  </Sender-List>

  <Recipient-List>
    <Agent ID="cAgtCtrTL-01" IsPrimary="True" />
  </Recipient-List>

  <Semantic-Content >
    <Ontologies-List>
      <Ontology Priority-Order="1">Traffic-Flow</Ontology>
    </Ontologies-List>
    <Context-Expression-Value>
      <Result scalar="True" datatype="Int">191</Result>
      <!-- Result from the request message -->

    </Context-Expression-Value>
    <Rules />
  </Semantic-Content>

</Agent-Message>

```

Code excerpt 5.2: Reply Message Example

There are different types of message instructions for representing the intention of the Agent in the communication layer. These message instructions are based on FIPA ACL performatives however the overall message structure is not FIPA compliant as the platform required more complex performatives than the FIPA provided. *Table 5.1* lists all the message types/instructions supported in the communication layer (in ITS@CU) for communication between Agents and highlights and justifies the improvements made.

Message Type	Description
Request	<p><i>Request</i> is the most common command used by an Agent to instigate a communication with another agent to request it to perform some action(s). It can be a simple Query or an Action request containing complex semantic contents.</p> <p>Optionally, conditions can be specified by the sender on which the request has to be fulfilled <i>e.g.</i> Perform action(s) only <i>if</i> or <i>when</i> condition statement is true.</p> <p>As compared to the FIPA specification which has <i>query</i> and <i>request</i> as two separate performatives (mentioned in “<i>Appendix D</i>”), the communication approach in this research merges both performatives in the <i>Request</i> in order to simplify the messages instruction without losing its functional strengths as the platform communication layer also include semantic-content (described in <i>chapter 6</i>) which provides the capability to embed complex queries/requests in the same message, so there was no need for separate <i>query</i> and <i>request</i> performatives/messages instructions.</p>
Reply	<p>In response to a <i>Request/Inform/Propose</i> etc., the recipient Agent uses a <i>Reply</i> command containing the possible outcome types (as Instruction-Type message):</p> <ul style="list-style-type: none"> • <i>Refuse</i> to perform the action with reason code(s) <i>e.g.</i> security/access level, role restrictions etc. • <i>Failure</i> or unable to perform the action with reason code(s) <i>e.g.</i> message corrupt, ambiguous ontology/rules, action not possible due to system problems etc. • <i>Response-Result</i> usually means successful response to the requested message. The response result is part of the <i>Semantic-Content</i>, and can be complex semantic instructions or basic scalar type of response such as True/False to confirm or disconfirm a statement by the sender. • <i>Agree</i> to indicate that the requested <i>Action</i> will be performed by the Agent (when possible or the condition is met). This is useful in situations where the reply message does not include a result or the request was conditional. • <i>Acknowledgement</i> is just to acknowledge that the message was received successfully. Every message must be acknowledged if requested by the sender. Usually other

	<p>reply messages are followed by acknowledgment.</p> <p>As compared to the FIPA specification, the <i>Reply</i> message in this approach is very flexible and incorporates various other commands as outcome types (Instruction-Type in message structure) within the <i>reply</i> message which are separate FIPA performatives (mentioned in “Appendix D”). As the platform communication layer also include the capability to embed complex semantics in the same message, so there was no need for separate specific purpose performatives/messages instructions.</p>
Cancel	<p>This is sent whenever an Agent wants to cancel a previously submitted message of any type. It is a simple message with <i>Cancel</i> message type with the Message ID value.</p>
Forward-Request	<p>It is similar to <i>Request</i>, but <i>Forward-Request</i> indicates that the message is forwarded on behalf of another Agent. This is important in situations where an Agent wants to relay Request messages to Agents not in direct reach or within its area of influence. The current sender just acts as a proxy to forward the message.</p>
Inform	<p><i>Inform</i> message is sent whenever an Agent wants to share information or notify another Agent (or Agents).</p> <p>This can be a simple notification based on an event or a change in the Agent’s status or situation. It can be message for a single recipient or a broadcast message for multiple Agents.</p> <p>It is used mostly during active Requests or notifying subscribed Agents.</p> <p>Agents can use either <i>Reply</i> or <i>Inform</i> in response to the message, if required.</p>
Forward-Inform	<p>Similar to Inform, however <i>Forward-Inform</i> indicates that the message is forwarded on behalf of another Agent. This is important in situations where an Agent wants another Agent to act as a proxy to forward its information.</p>
Subscribe	<p>It is a message sent by an agent to another agent if it wants to be notified whenever some event occurs. To unsubscribe, the same message with Unsubscribe Instruction-Type value as True will stop further notifications.</p> <p>Agents use <i>Reply</i> to confirm subscription/un-subscription status (<i>Response-Result</i> Instruction-type with value of True)</p>
Register	<p>An Agent sends this message to register/advertise its capabilities. This is mainly used by Service Agents; however Operational Agents can also use it for similar purposes.</p> <p>The same message with Unregister Instruction-Type value as True will cancel the</p>

	<p>registration.</p> <p>Similarly, the Agent uses <i>Reply</i> to confirm registration status (<i>Response-Result</i> with value of True in the reply message)</p>
Propose	<p>This message is used in negotiation between agents proposing actions e.g. an Agent can propose another Agent to apply/consider a specific set of actions as a solution to a problem or deal with a particular situation.</p> <p>Agents use a <i>Reply</i> message to respond to a proposal for example Rejecting the proposal or requesting more proposals</p>

Table 5.1: Message types/instructions in the agent communication layer

All the messages were defined in XML format which offers a number of advantages:

- XML is platform independent so is suitable for use in heterogeneous environments. XML is standardised by W3C and it is interchangeable between systems and programs which is key for heterogeneous entities needing to interact with each other (w3c.org).
- Compared with other encoding formats (e.g. text file, RDF, XHTML, JSON etc.), XML is easy to generate, parse, edit, and translate because of its strict and consistent syntax, and high standard of XML tools are available for these purposes (Chen et al., 2008; Hameseder et al., 2011).
- XML allows defining of custom document structures, which is ideal for Agent based communication ontologies (Daigneau, 2011).
- Native support for .NET which is the core technology used for ITS@CU platform development (Erl et al., 2010)
- XML can be bound and validated against a schema which was important for Agent based messages to conform to a common structure (schema presented in *figure 5.3* and “*Appendix M*”)

The following is the XML Schema Document (XSD) to define the overall messages structure:

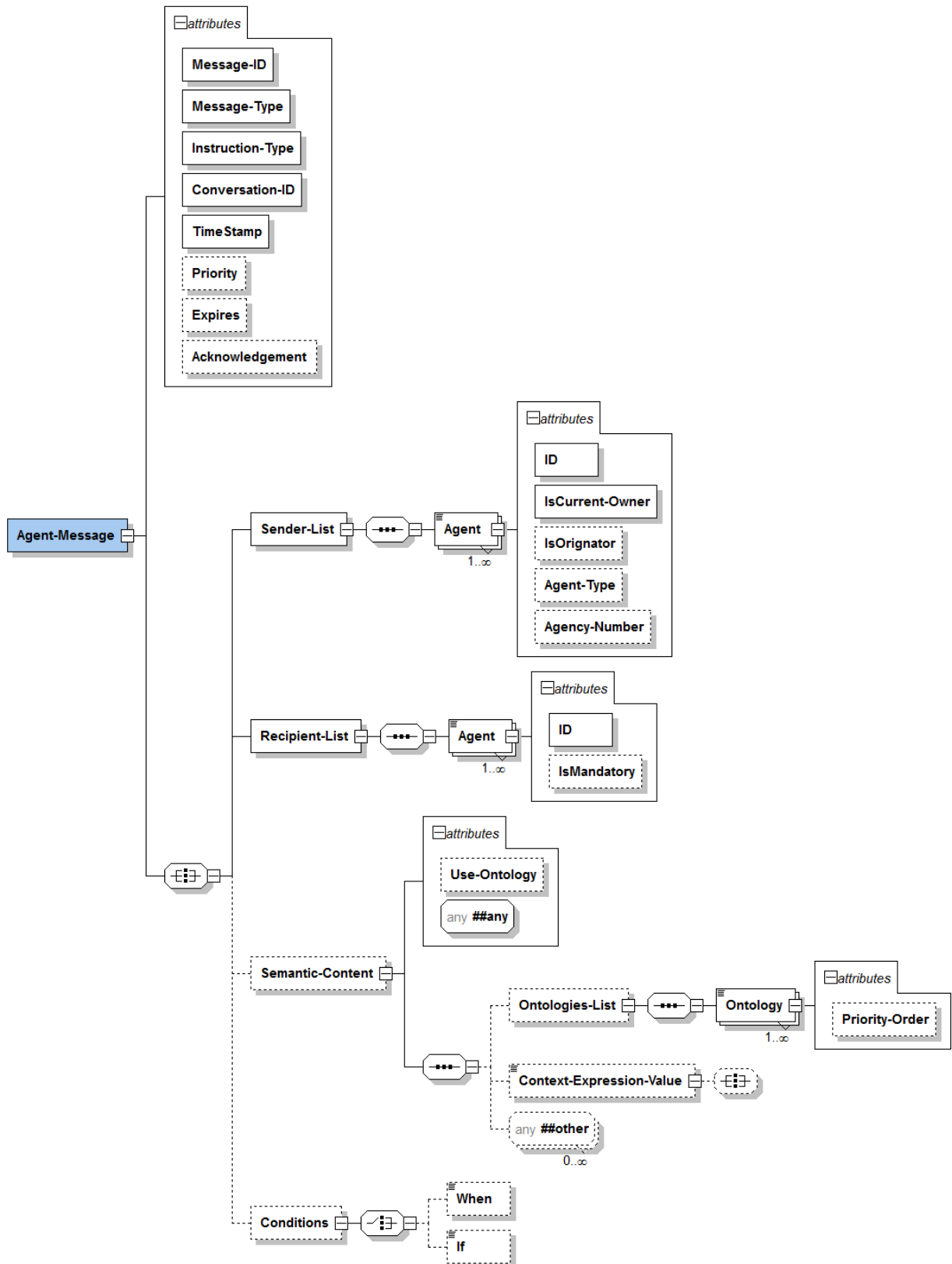


Figure 5.3: Agent Message structure (XSD Schema)

The complete detail of the elements and attributes of the message structure are covered in “Appendix M”.

5.3. Agent communication lifecycle

The multi-agent communication approach adopted in this research can be just a one-off request and reply message between two Agents or it can be quite complex where communication involves a chain of messages passed between multiple Agents multiple times. As an example, consider a situation where a part of a road becomes congested in ITS@CU configuration, the Control Agents for vehicle count sensors controllers on the congested road section will report above average flow/occupancy of vehicles to the relevant Link Control Agent, which then requests an Arbitrator Operational Agent to propose a diversion or any other action(s). The Operational Agent then liaises with multiple other Agents including Service Agents if a diversion route is required. Based on the outcome, the Operational Agent sends a proposal or request to the affected Control Agents including the originator Control Agents.

See *figure 5.4* showing the overall communication flow for the above described example scenario

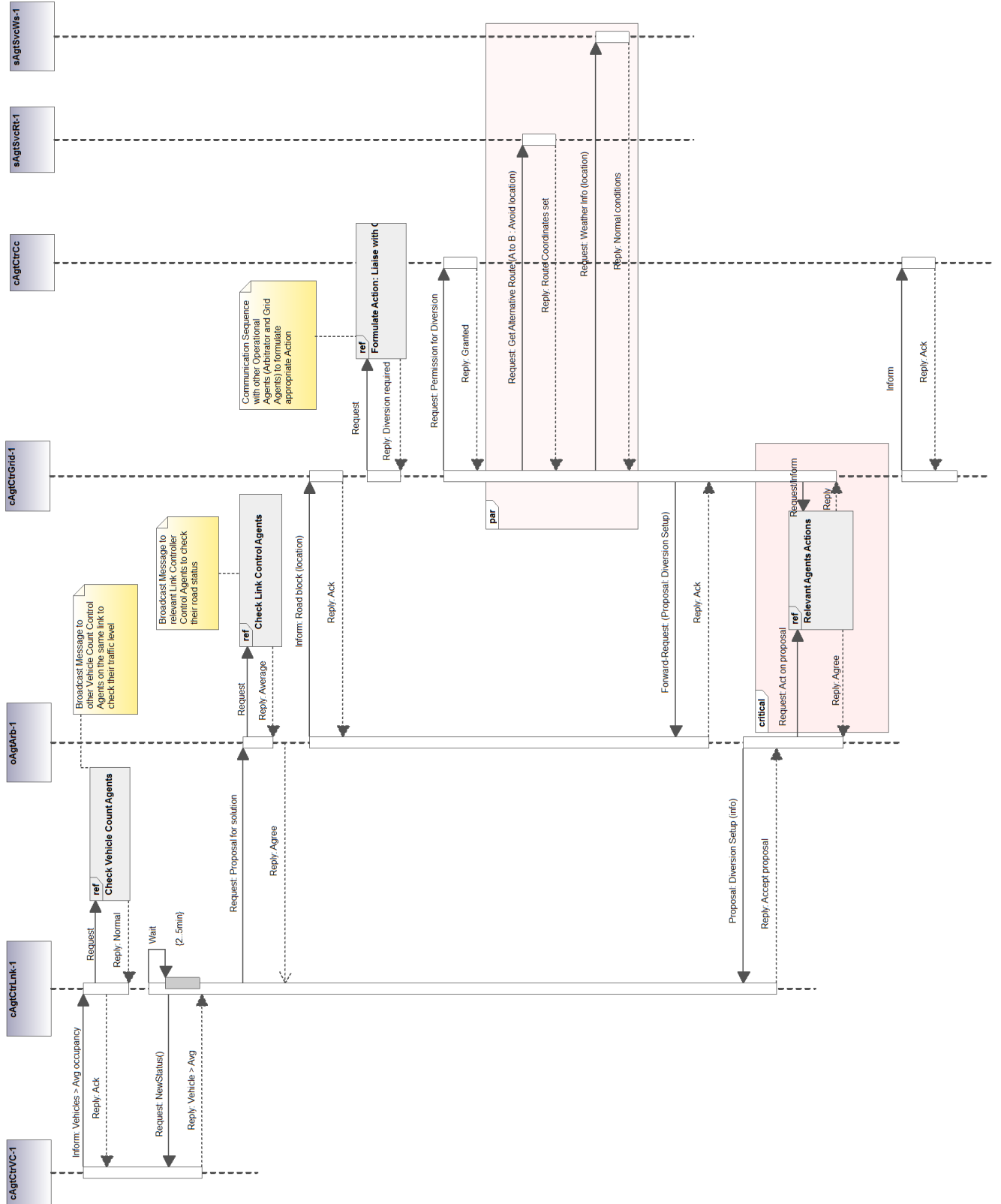


Figure 5.4: Example Communication flow (Multiple Agents and Message types)

Every message has mandatory attributes “*Message-ID*” and “*Conversation-ID*”. The *Message-ID* uniquely identifies the message and it is generated as a Globally Unique Identifier (GUID) every time any message is generated between any Agents. A *Conversation-ID* uniquely identifies the message thread between

multiple Agents which originated initially from a single message. The Conversation-ID is also a GUID to avoid any ambiguity.

The conversation flow therefore has a defined start and end-point. A conversation is ended either by the originator sending an *Inform* or a *Cancel* message. Alternatively an Operational Agent such as an Arbitrator Agent can end the conversation, if necessary.

Multiple Messages can be also bundled by the message interface module however each message still maintains its complete structure with a unique message-ID to keep messages separate and avoid any ambiguity.

Messages are parsed by a “Message Processor” which is part of the Interface Manager (shown in *figure 5.1*). It separates the message instructions and embedded semantic-content for an Agent to perform further analysis.

Chapter 7 covers further details at the implementation level.

5.4. Security

Security is vital for any system and network, however agent based systems require an extra level of security as compared to other distributed technologies as an Agent is autonomous and can cause intentional or unintentional harm to a system. It may affect the performance of the host, communication channel or compromise data validity and privacy. In the agent communication layer, various measures were adopted to implement security such as:

- Every Agent is uniquely identifiable and has a specific role and level of access. For example, a Control Agent can only send *inform messages* to other Control Agents and *request* limited actions, however a “Controller Manager” Operational Agent can create, suspend and access multiple Control Agent’s data.
- Security is handled by a specialised “Security” Operational Agent which verifies Agents and their messages according to their roles, level of access/authority, trust and usage allocation policy. If an Agent requests another Agent, the message is verified by a Security Agent to check if the Agent is trusted and has the level of authority for the actions it is requesting.
- During the hosting mechanism the Control Agents use only the agreed level of host resources

specified as a usage policy agreement by the “Controllers Manager” Operational Agent in order to avoid the excessive exploitation of the host CPU or memory.

- The relevant Agent Manager Operational Agents (e.g. Control Agents Manager for Control Agents) can stop or suspend an Agent, if required. The “Arbitrator” agent can also cancel a conversation or message request.

Chapter 7 covers further details at the implementation level.

5.5. Conclusion

This chapter described the agent communication layer and the message instruction types used by Agent-based Controls and other platform Agents to efficiently communicate and coordinate with each other. Agents communicate using messages based on a unique structure composed of an outer message instruction part and a semantic-content part. The chapter covered the types of message instructions/commands, which were based on FIPA performatives but adapted with improved functionality due to the enhanced capabilities of the semantic-content supported in the agent communication layer.

The next chapter provides further details about the “semantic-content” part of the Agent message which is the essential part of Agent communication for understanding the agent’s intentions in order to work together to achieve their goals.

Chapter 6

Agent Semantics, Ontologies and Rules structure

The previous chapter presented the structure of the Agent communication layer for multi-agent communication and coordination in the ITS@CU platform.

This chapter is focused on the design of the Ontologies and Rules used in multi-Agent communication to describe the semantic behaviour of the Agents and facilitate in decision making. The chapter has the following sections:

- The first section describes the steps or flow of Agent messages (from receipt to action)
- The second section describes the structure of the “semantic-content” part of the Agent messages
- The third section describes the structure of the Ontologies associated with Agent messages
- The fourth section describes the structure of the Rules, and selection and matching criteria used by agents to formulate Plans as part of the decision making process
- The fifth section outlines the formulation of Plans by agents based on the Rules in order to respond to situations/perform actions

6.1. Agent message flow

Agent-based controls require ontologies for understanding the semantic-content in agent messages and to realise the communication intention. Ontologies semantically describe the Agents' behaviours and coordination for efficient communication. When Agents communicate about a domain then it is necessary that there is an agreed set of common terminology that these agents understand for describing that domain.

This section describes the core structural design of ontology and rules set which can be embedded as part of the Semantic-Content of the agent message.

Figure 6.1 presents the steps which an agent message goes through in order for an Agent to perform the required actions.

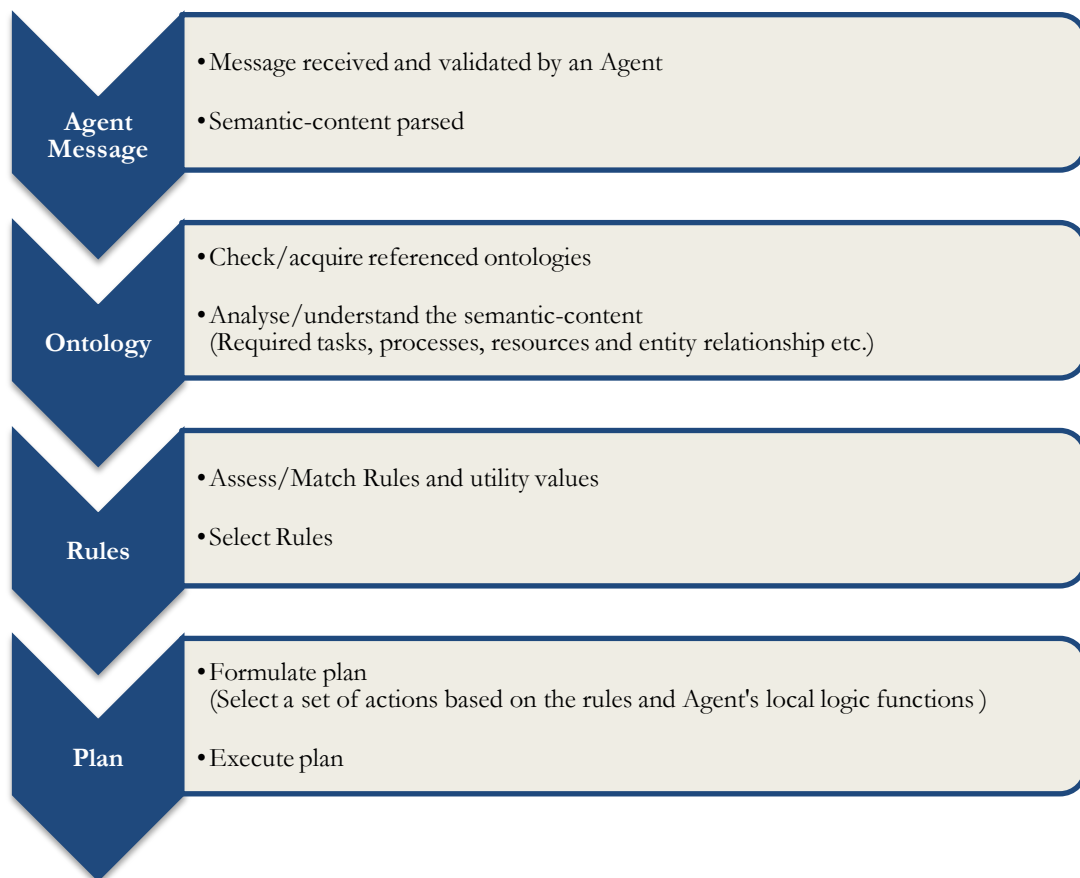


Figure 6.1: Message flow overview

Ontologies and rules can be stored in files or in supported databases as an XML data type. The storage depends on the type of host e.g. Grid controller can host a light version of DBMS (SQL Server Compact Edition which supports the XML data type), but a vehicle count sensor Controller can only store basic XML files.

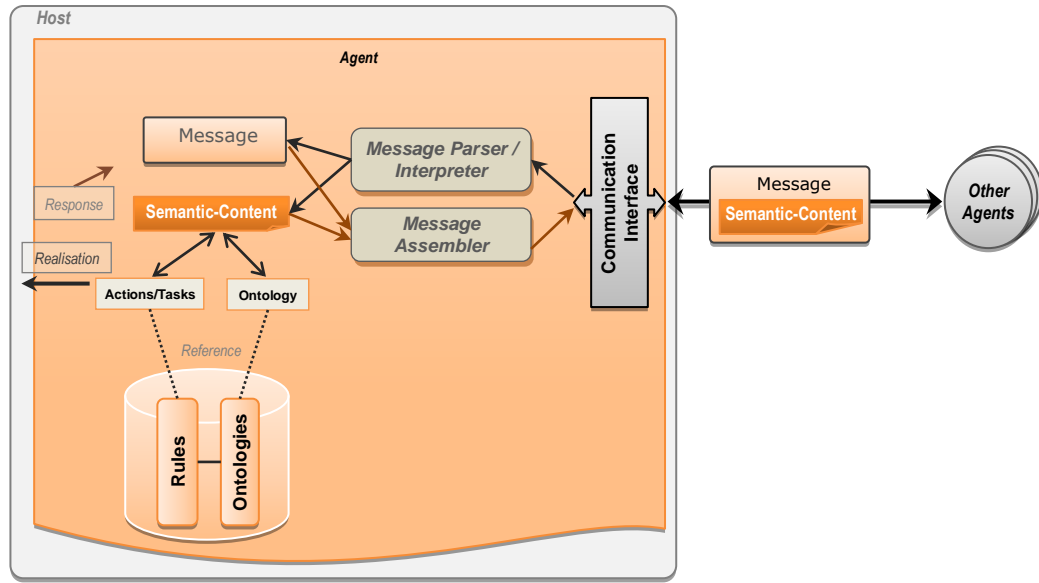


Figure 6.2: Message & Semantic-Content composition and parsing

As seen in *figure 6.2*, an Agent receives a XML message comprising of message transport instructions and semantic-content. It is parsed by the interpreter/parser module which extracts the semantic content from XML document tree by splitting the message (see example *code excerpt 6.1*). The Agent then analyses the semantic-content part in accordance with the referenced ontology and also if any corresponding rule set can be applied from the rule set in its local memory storage.

The semantic-content of an agent message can be a simple request such as the example code excerpt below or it can be very complex with references to multiple ontologies and new rules embedded within the semantic-content part. If the agent does not have the referenced ontology or has only a partial ontology then it composes another message to request the ontology from the requester agent or an operational agent. Similarly, if it does not have any rule or partial rule set it can also request the requester agent or a relevant operational agent to send matching rules. In this way the agent can perform a very complex task in coordination and cooperation with other agents. This also means that the agent does not need to have all the ontologies and rules available locally as it can always obtain them when required. This helps in keeping the local stored information to a minimum on the host hence reducing the need for more storage and memory. Although requesting ontologies and rules can result in a communication/data overhead, its benefit far outweighs this issue.

Once the Agent has the complete ontology and rules required for the message semantic-content realisation, then it can perform the required task(s).

Similarly, if the agent wants to send a message it composes the message and embeds semantic-content based on rules and ontology. The message is composed by a “Message Assembler” module which generates a XML document message file according to a valid schema (See *figure 6.3* and the details in “*Appendix M*”).

```

<Agent-Message Message-Type="Request" Instruction-Type="Action"
  Message-ID="MSG00000123" Conversation-ID="CON-MSG00000123"
  TimeStamp="2010-11-11 19:23:44" Expires="2010-11-11 19:25:44">

  <Sender-List>
    <Agent ID="cAgtCtrTL-01" IsOriginator="True" IsCurrent-Owner="True" />
  </Sender-List>

  <Recipient-List>
    <Agent ID="cAgtCtrLnk-07" IsPrimary="True" />
  </Recipient-List>

  <Semantic-Content >

    <Ontologies-List>
      <Ontology Priority-Order="1">Traffic-Flow</Ontology>
      <!-- Can be multiple Ontologies and more attributes such as context, location, version etc. -->
    </Ontologies-List>

    <Context-Expression-Value>
      <!-- The rest is only understood by the relevant Recipient -->
      <Get-Link-Occupancy filter="">
        <Link-Name>FolesHill Road</Link-Name>
        <Start-Nodes>CtrNd-005</Start-Nodes>
        <End-Nodes>CtrNd-006</End-Nodes>
        <!-- Can have various other elements/Attribute based on ontology -->
      </Get-Link-Occupancy>
      <!-- Can have various other elements/Attributes -->
    </Context-Expression-Value>

    <Rules /> <!--Rules relevant to the message, if required -->

  </Semantic-Content>

  <Conditions />
</Agent-Message>

```

Code excerpt 6.1: Example XML Message and Semantic-Content

6.2. Semantic Layer (Semantic-Content) description

As seen in *code excerpt 6.1*, the element “Semantic-Content” within the “Agent-Message” XML message document contains the semantic information required for the recipient Agent to analyse and process. It is not bound to a strict XML schema as compared to the message itself. The main message XML schema uses `<any>` element and `<anyAttribute>` attribute to allow flexibility in the semantic-content part of the XML document (see *figure 6.3*).

In the semantic-content there are few common elements and attributes shared by all agents and the rest of the nodes are specific to the type of Agent. The common nodes/elements are:

- `<Ontologies-List>` List of ontologies required as a reference to understand the request

message. It also includes the priority and URI (location if the recipient agent wants to retrieve the ontology).

- **<Context-Expression-Value>** Context expression containing the actual tasks or actions required. This is the core part of the entire message and is designed to be flexible allowing any type of attributes and elements.
- **<Rules>** List of Rules which the sender Agent sends to the recipient in order to deal with the required tasks. It is up to the recipient agent to decide whether to apply the rules or not.

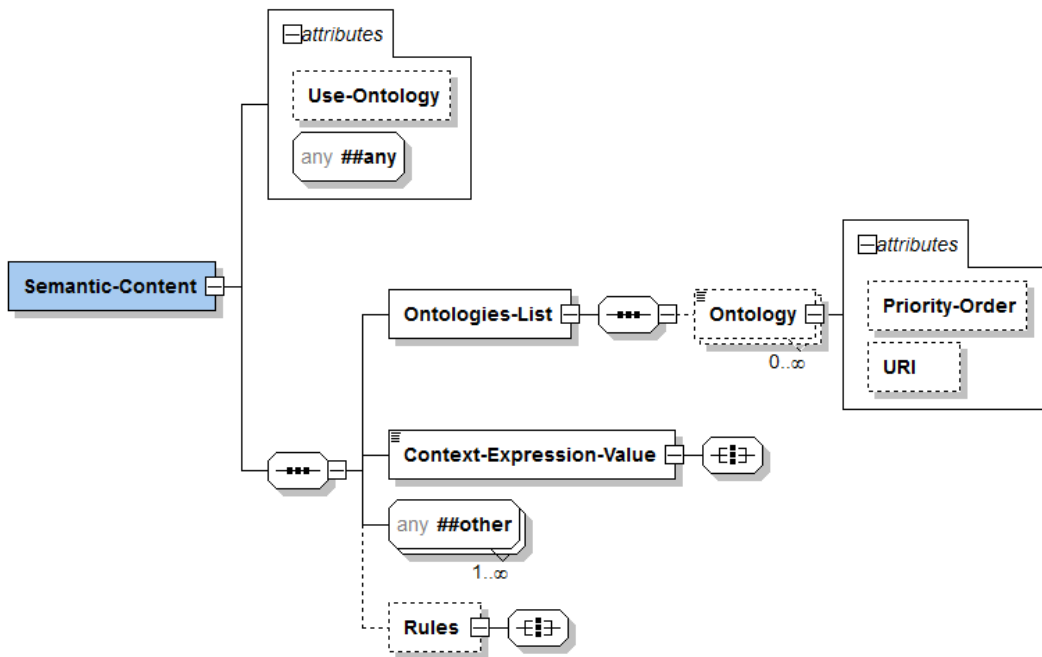


Figure 6.3: Semantic-Content structure (XSD Schema)

Further details of all the elements and attributes are covered in “Appendix N”

The example *code excerpt 6.1* shows simple semantic-content within a message between two control agents. The recipient, upon receiving the message, interprets the **<get-link-occupancy>** node and refers to its ontology to understand what action is required. The ontology “traffic-flow” (see *figure 6.6*) has the definition of get-link-occupancy and its sub nodes. After referring to the ontology it will ascertain that the request is about the current occupancy of a link between certain junction nodes (as per ontology relationship in *figure 6.3* and terms explanation). Once the message is interpreted, the recipient agent will check against its rule set for an appropriate response. In this example no rules were provided by the sender implying that the recipient agent has the appropriate rules to deal with the request.

In response the agent composes another message and embeds the return value in the semantic-content (as

shown in the *code excerpt 6.2*), also referring to the same ontology. Further details related to this message and ontology example are covered in the next section.

```
<Semantic-Content >
  <Ontologies-List>
    <Ontology Priority-Order="1">Traffic-Flow</Ontology>
  </Ontologies-List>
  <Context-Expression-Value>
    <Result scalar="True" datatype="Int">191</Result>
  </Context-Expression-Value>
  <Rules />
</Semantic-Content>
```

Code excerpt 6.2: Example Semantic-Content in a reply message

6.3. Ontology structure

The approach adopted in ITS@CU includes various ontologies for Agent communication. The ontologies are classified as top-level, domain-level or task-level:

Top-level: Describes general concepts and metadata in overall ITS@CU platform, for example, resource, roles and agents types and basic message terms/concepts. In this research, both foundational and generic ontologies are treated as top-level ontologies. The reason for adopting this approach was commonalities between the two in the platform and also to simplify/promote basic understanding among the agents.

Domain-level: Describes the terminology specific to a particular domain for example the signal adaptation ontology is understood at local traffic light controllers by the involved agents. It is also used to further elaborate the top-level ontologies according to the domain.

Task-level: These ontologies are specific to a particular task or process for example traffic-flow optimisation, broadcast alerts on route, perform recovery task etc.

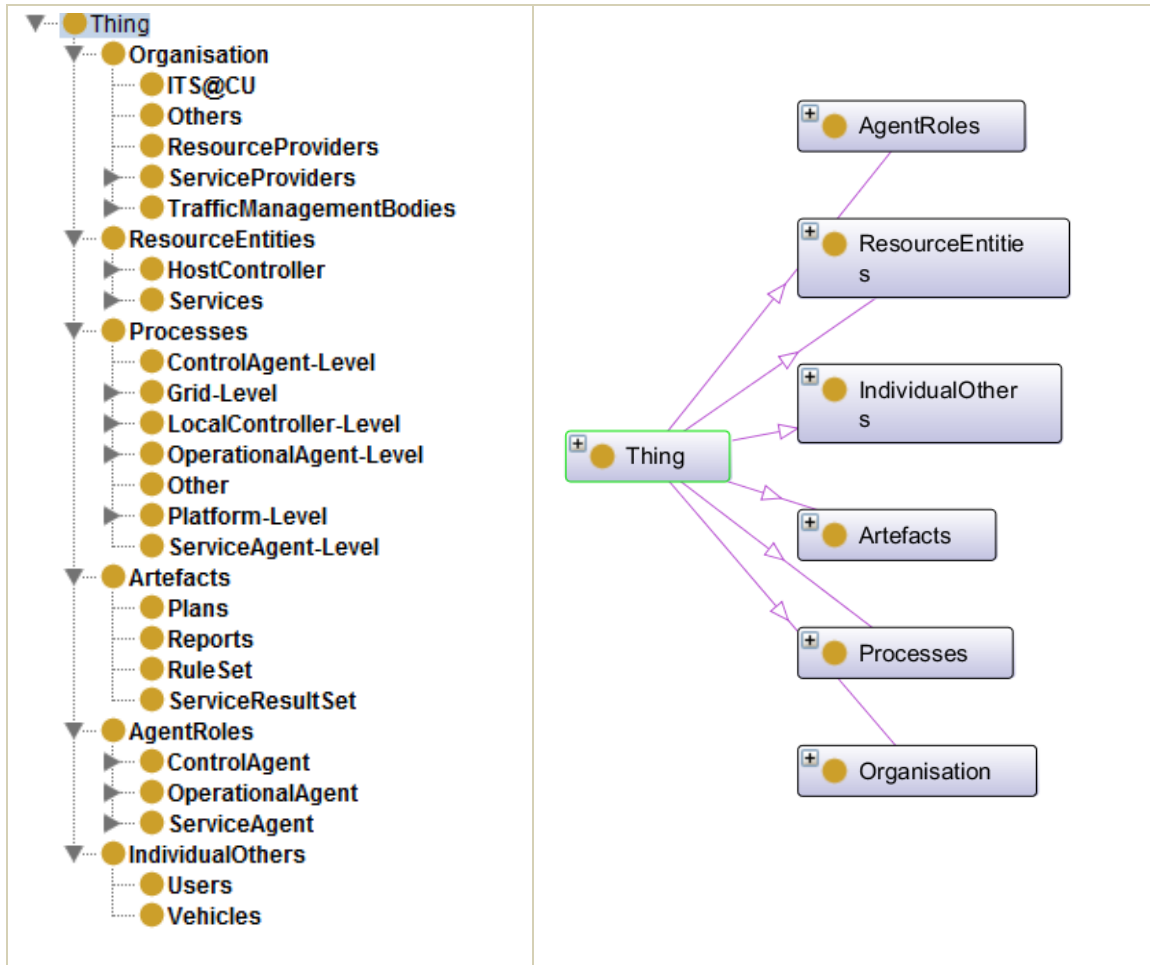


Figure 6.4: Top-Level Ontology Structure (classes)

As seen in *figure 6.4*, the overall ontology structure consists of the following type of classes:

Organisation: Service providers for example Microsoft Bing which is providing the route service is a separate organisation in the description of ITS@CU’s top-level Ontology.

Resource entities: Describes the resources (host controllers or Services) used in the processes for example Route Service, local traffic controllers/devices, central control system and other such host systems that can be used as resources in the platform system.

Processes/Tasks: Describes the actual processes involving a set of tasks for example “signal adaptation” process for traffic lights Control Agent, “Broadcast” specific message by operational agent to relevant recipients etc. Control Agents rely on these ontologies for its decision making process.

Artefacts: Describes the output result (product) of a process, for example “Divert traffic” process results in a plan or plans for Control Agents to implement, “Report current vehicle flow” process generates a report of the current status of the vehicle flow on a specific road or section.

Roles: Describes the Agents roles responsible for executing tasks/processes and providing resources/services, for example, “Traffic Lights Control Agent” performs the role of managing/controlling set of traffic lights controller, “Route info Service Agent” provides route and traffic information, “Security Agent” checks message validity and authenticates agents acquiring a resource.

Individuals/Others: Additional classes which are not directly associated with above classes for example vehicles. They are however interlinked with the above classes and use the ontologies in conjunction with other classes.

Agents use ontologies at different levels to understand the semantic contents of the messages. Ontologies not only provide the relationship and understanding between entities, resources and their properties, but also references to rules within processes which enables Agents to take appropriate and circumstantial actions. This is especially important for Control Agents mainly due to the dynamic nature of Control Agents and also the limitations of storage/memory (on its host). A Control Agent relies on dynamic ontologies for its operations rather than storing all rules and functions locally on the traffic Controller host.

The following *figure 6.5* shows the complete top-level ontology used in ITS@CU. The top-level ontology is further elaborated and fine grained using set of domain and task-level ontologies.



Continued on the next page



Figure 6.5: Top-level/main ontology (used the ITS@CU implementation)

The following *figure 6.6* is a fine grain example of a task and domain level ontology derived from the top-level ontology. It describes a simple “process” ontology which is applicable to Control Agents within a grid and local traffic controller’s domain (Traffic Counts Control Agents in particular).

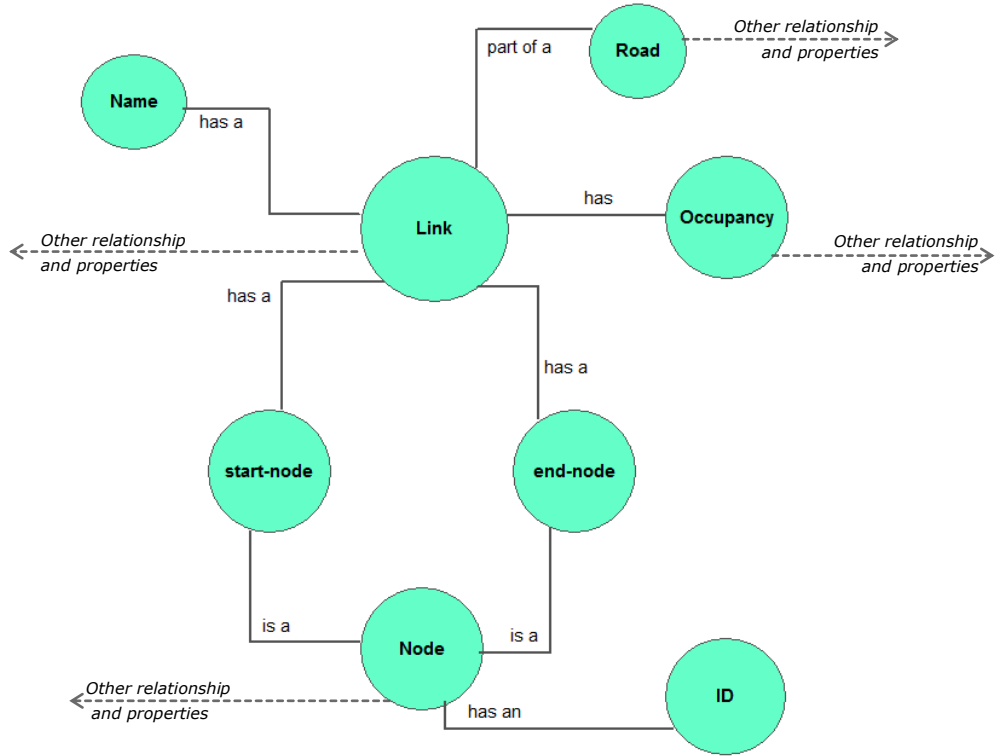


Figure 6.6: Part of Ontology “Traffic-Flow”

In the previous example mentioned in *section 6.2* (and *code excerpt 6.1*), the semantic-content refers to “traffic-flow” ontology (partially illustrated in *figure 6.6*). The receiving Control Agent of the message will look up (or acquire) this ontology to understand and analyse the semantic content under the context expression element. Using this ontology the control agent can understand what a link-name refers to, and what a start and end-node means. It will also understand the function required to fulfil the request i.e. get the current occupancy of the road or link-name mentioned in the message. The functions would be matched in the local rules in the Agent’s memory or storage. Using the relevant rules the agent will formulate a plan of action to execute the request. Rules and plans are discussed in the next section.

Similarly, there are various other task and domain-level ontologies which are used in ITS@CU for agent based communication (ontologies included in “*Appendix J*”).

Ontologies can be defined in various formats such as RDF and OWL (mentioned in *section 2.3.3*), however in this research, XML was used for defining ontologies and rules due to its flexibility and suitability of SOA based environments.

6.4. Rules structure

Rules describe the beliefs of an agent in the context of the agent implementation in this research. Rules are fundamental for Agent-based Controls and forms part of the Agent-Logic and can be also embedded in semantic layer of agent messages (see *figure 6.3* and the details in “*Appendix M*”).

Match-Rules:	
R-1:	IF \$user-info\$ <i>i</i> in range of \$Condition-Value-Set\$ && <<select-rule-R5>> has goal <i>G</i> THEN <<select-rule-R6>> with parameter <i>i</i> , <i>G</i>
R-2:	IF \$Agent-name\$ <i>Agt</i> is part of >>check-Agency-number(<i>Agt</i>)<< && \$Controller-ID\$ <i>ctr</i> is part of <i>Agt</i> THEN <<allow-actions>> on <i>ctr</i>
R-3:	IF >>user-action<< <i>ac</i> = Complete <i>ac</i> = \$defined-value\$ THEN <<select-rule-R9>>
R-4:	IF \$defined-value\$ <i>value</i> in message is ambiguous undefined THEN >>generate-response-ontology(<i>value</i>)<<
Select-Rules:	
R-6:	IF <<select-rule-R6>> THEN Wait %set-wait-time% && >>generate-reply-accept<< with parameter >>generate-reply-accept<< in %set-wait-time % IF response \$status\$ = high THEN >>generate-reply-accept<< to <i>aAgtArb-1</i> for >>generate-reply-accept<< with parameters <i>i</i> , <i>G</i>
R-7:	IF <<select-rule-R7>> THEN >>send-request-stop(\$controllers\$) AND >>Change-status(\$defined-value\$)<<
R-8:	IF <<select-rule-R8>> THEN >>get-link-occupancy(\$start-node\$, \$end-node\$, \$link-name\$)<< From \$Agent-name\$

\$value\$	= variable values
%attribute%	= local attribute value of the Agent
<<select-rule>>	= Reference to other Rules
>>local-function(<i>parameter</i>)<<	= local method of the Agent

Figure 6.7: Rules structure example (of a Control Agent)

Rules are a set of match and select cases with possible response actions which an agent uses for its operations and to deal with events. As seen in the rules structure example in *figure 6.7*, the rules are of two types (i.e. Match-rules and Select-rules) which an Agent uses for its decision making process and to

formulate a plan of actions. An Agent uses match-rules in order to check if any condition or request by another agent matches an already existing rule. Match-rules can refer to select-rules which are action oriented and important for an agent to create a plan of action(s).

Agents can change/update their Rules if requested by other Agents with authority (such as Control Agents Manager). Any such update comes as a request message. However not all rules are updatable by other Agents as they are a fixed type of rules for example rules in the default-logic of a Controller. The update mechanism provides flexibility and allows the traffic controllers to be updated whenever new and better functionality or rules are available (released by Manager operational Agents).

Rules are defined in XML and stored in the agent's local data store (host database or xml file), see “*Appendix J*” containing all the rules which were used in ITS@CU implementation and evaluation purposes.

6.5. Agent Plans

Once an Agent identifies the matching rules and appropriate select rules (in response to the situation/event it is dealing with), it dynamically formulates a plan of action (based on these rules and their agent-logic operations). The plan contains various action steps with different utility values. Utility values are used as weighing factors for the selection of an individual plan's action. An action is treated as a goal for an agent to achieve and the agent aims to perform the action with highest utility value possible.

The following figure 6.8 shows an example of plan structure.

```

GOAL:
  RESTORE-TRAFFIC-FLOW

SITUATION
  Road block "Link-1" between "Node-05" and "Node-06"
  aAgtArb-n confirmed diversion
  sAgtSvcRt-n provided alternative route

Plan-1:
  UTILITY: 20
  <<select-rule-R6>>
  THEN >>generate-inform-message<< to aAgtArb-n
  FAILURE: Retry $set-value$ times ELSE execute Next higher utility value plan

Plan-2:
  UTILITY = 10
  IF checkFlowStatus(route-1) from aAgtArb-2
  <<select-rule-R-20>> AND <<select-rule-R-10>>
  THEN >>generate-inform-message<< to aAgtArb-n
  FAILURE: Plan3

Plan-3:
  <<select-rule-R-0>> -- do nothing
  THEN >>generate-inform-message<< to aAgtArb-n

```

Figure 6.8: Example structure of a plan (produced by a Control Agent)

Plans are also defined in XML, See “*Appendix J*” containing various examples of Rules used by Agents in ITS@CU platform.

6.6. Conclusion

This chapter presented the approach of Ontologies and Rules used in Agent communication to describe the semantics of Agents’ messages and facilitate decision making. It outlined the flow of an Agent message in this platform which goes through the following steps: message interpretation; Ontology processing for understanding the message semantic-content; Rules selections and matching to formulate a plan of action.

The semantic layer approach in Agent communication provides a high-level of flexibility in the form of semantic-content containing expressions, Ontologies and Rules embedded within Agent messages. In this way the agent messages can contain sophisticated semantic-content commands with references to multiple ontologies and rules at various levels. This provides Agents with a communication framework to work in synergy and coordinate distributed decision making at different domain levels.

Chapter 7

Implementation of ITS@CU platform

The previous chapter described the design of the Ontologies and Rules used in multi-Agent communication to describe the Semantic behaviour of the Agents and facilitate in decision making.

This chapter presents the design and development description of the ITS@CU platform relevant to the concepts described in *chapters 4, 5 and 6*. It is divided into the following four sections:

- The first section provides an overview of the platform and its purpose
- The second section presents the architecture and design of the overall platform
- The third section covers the technical description of the platform's three main components, applications, services, databases and communication structure. It also highlights the tools and technologies used for the development of the platform.
- The fourth section outlines the Mobile Application Development Framework (MADF) which was developed as part of the research for the rapid development of different Controller and Control applications in order to simulate their behaviour.

7.1. Platform overview

The ITS@CU platform is the combination of systems/applications, services, development framework and communication layer modules implemented to meet the research objectives. ITS@CU was implemented using the novel concept of Semantic Agent-based Controls and by uniquely combining Multi-Agent, SOA, semantic web services and intelligent control agents with the aim of reducing the complexities and inefficiencies resulting from the integration and communication between distributed and multi-domain controls associated with ITS.

During the research, various platforms and systems (mentioned in *chapter 2*) were evaluated and studied; however each lacked certain core features needed to fully meet the research objectives and the commercial requirements. Additionally, there was a lack of suitable development platforms available for the simulation of ITS controls and rapid application development on a commercial level. Most of the simulation tools and platforms currently available are either too specific, technology centric or in early research stages.

The ITS@CU platform was therefore developed from the ground up in order to achieve the following core objectives and requirements:

- Ability to support the Semantic Agent-based Controls concept (presented in *chapter 4*).
- Support dynamic cooperation and coordination between different Agents, services and systems from different domains and grids using the semantic data/ontologies, and respond to traffic situations using the rules and plans (described in *chapter 6*).
- SOA principles based communication and integration architecture to support multi-Agent communication (described in *chapter 5*) using web services in a highly distributive environment.
- Capable of integrating with other commercial systems and using commercially proven tools and technologies.
- Provides features for the rapid development of applications to simulate various Agent based Controls (Traffic Controllers) and roadside controls.
- Provides supporting utilities and applications for the management and simulation of the system and its components.

7.2. Design and architecture

7.2.1. Platform components

Modern traffic networks consist of various systems and components for example roadside devices/sensors, regional/central management system, vehicles, traveller information and flow control equipment. In order to design and develop the ITS@CU platform (and associated sub-systems) it was divided into three core components:

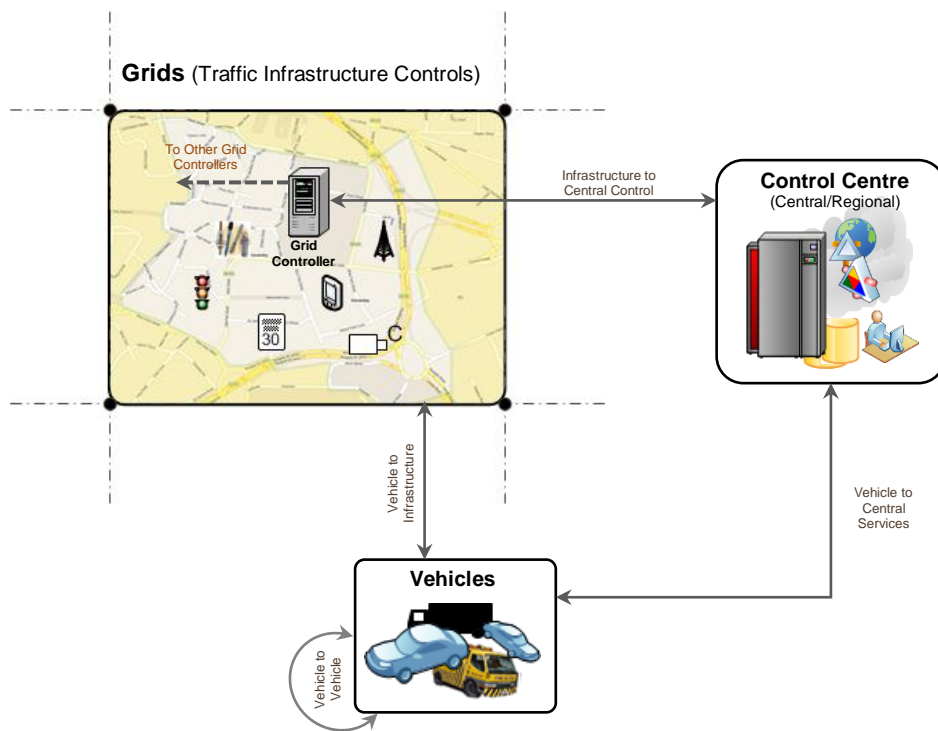


Figure 7.1: Components of the ITS@CU platform

Central Control System: This component consists of a central system (representing a set of system tools/utilities) responsible for managing the entire traffic infrastructure i.e. Control Agents, grid controllers and vehicle controls. It provides the central point of communication and coordination for grid controllers and operational Agents. Although grids (and systems within grids) form a network of distributed systems with internal decision making power, it is necessary for security and safety reasons to have a Central Control which arbitrates and controls the grid and control agents at a higher level.

Grid/Traffic Infrastructure Controls: This component includes all the traffic controllers (traffic lights, sensors, dynamic message boards/signs etc.) which are part of the road network. Based on the grid approach described in *chapter 3*, traffic infrastructure is divided into smaller variable size zones/areas.

Each grid is managed by a “Grid Controller” which hosts various types of Agents having different roles responsible for controlling and managing all the traffic Controllers (and Agent-based Controls) within the Grid area and communication with the control centre, vehicles and other grid controllers.

Vehicle Control System: As vehicles are dynamic entities they cannot strictly be part of any infrastructure grid controllers or in direct control of the infrastructure controls. However, in ITS based environment it is vital that the vehicle communicates with other components/controls. In the ITS@CU platform, Vehicle controls are designed to retain full control but are able to communicate with the infrastructure controls and other vehicles in range.

7.2.2. Platform components design architecture

The ITS@CU platform is a distributed and multi-layer system comprising of various components, modules and services. For this reason, and due to the other benefits described in *chapter 2, section 2.5*, Service Oriented Architecture (SOA) was adopted as the core architecture for the overall integration and communication of the ITS@CU platform.

Each system or module within the platform has a different architecture for example a web application internally uses Model View Presenter (MVP) design architecture and the mobile application for Controllers uses component and object oriented design. The components are integrated in a loosely coupled manner using services and SOA principles i.e. each controller, system/sub-system within the platform is either a service or provides a service interface to others. SOA provides simple services based communication and integration between systems/components regardless of the systems’ underlying technologies.

The platform was designed to be generally de-centralised however the central control (Cc) system provides arbitration and governs the high-level rules to ensure security and safety. The more centralisation is increased the more it delays decision making as central control has to communicate with all the involved controls to make a decision. On the other hand, if the system is completely decentralised to agent/grid controls level then the traffic Controllers require more processing power and memory. The ITS@CU platform follows a hybrid approach where the central system arbitrates at a higher level and agent-based controls perform tasks in association and cooperation with each other.

As seen in *figure 7.2*, the platform components have different modules and layers. Each module performs a specific set of functions and comprises of various applications and utilities.

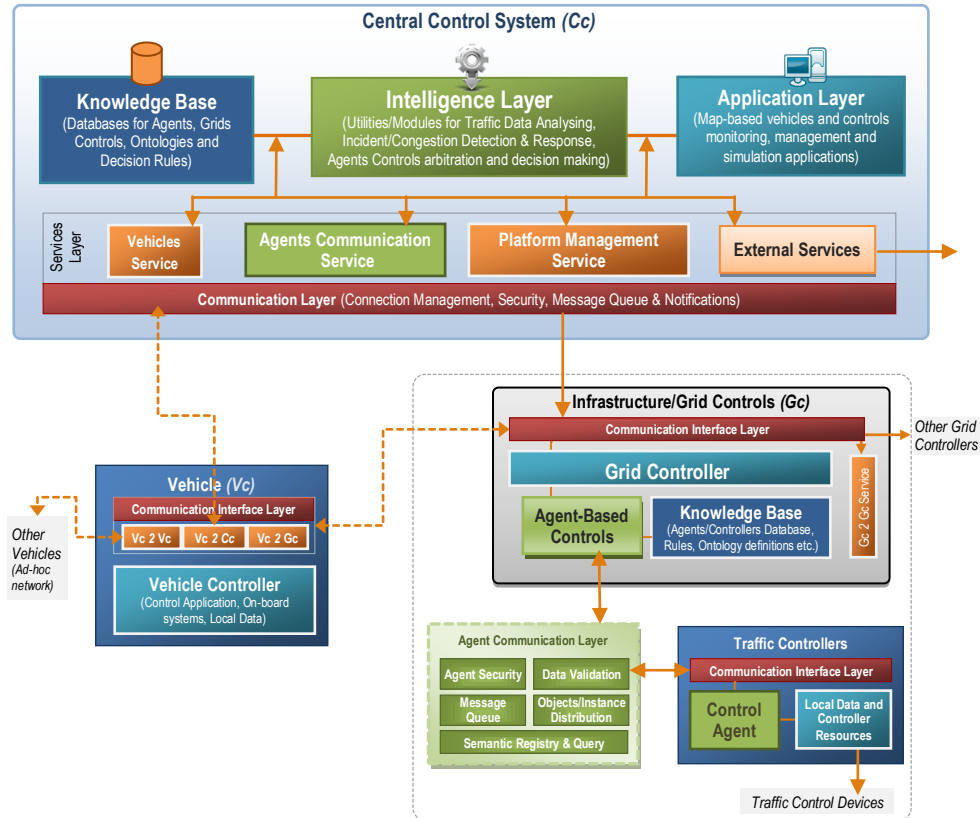


Figure 7.2: ITS@CU Architecture

The Central Control (Cc) system comprises of the following core modules and layers:

Services Layer	<p>Comprises of different services providing various functionalities to other components of the platform. The “Vehicles Service” is used by vehicle controllers to send location data (GPS) periodically which can be used for vehicle tracking and other traffic data analysis purposes. The “Platform management service” is used by grid controllers to send and receive traffic controllers and Agents related updates and administration instructions.</p> <p>The “External services” are used by Service Agents to access services from external sources (such as Bing Map route information or Met Office weather information).</p> <p>The “Agents communication service” is the core communication service interface used by Grid Level Agents to interact with Central control system level Agents.</p>
Communication layer	<p>Provides all the connection management functions (Message Queuing and Session management), Security (Encryption and Authentication) and Notifications to other components etc.</p>

Intelligence Layer	<p>This layer comprises of an application which perform vital functions such as Traffic Data Analysing, Incident/Congestion Detection & Response, Agents Controls arbitration and decision making.</p> <p>Additionally, it hosts Operational Agents (Central Control Level) and Service Agents for providing access to external service.</p>
Application Layer	<p>Includes web-based applications for vehicles and controls monitoring and management. The monitoring application has an interactive map based user interface. Additionally, it has various utilities for simulation purposes.</p>
Central Knowledge base	<p>Comprises of databases for Grids Controllers/devices, Agents and Ontologies/Rules.</p>

Table 7.1: Central Control modules and layers

The Grid Control (Gc) system comprises of the following core modules and layers:

Grid Controller	<p>This is an application for controlling and managing the Grid and associated traffic controllers. It provides the ability to host Agents and is responsible for the decision making/arbitration capabilities at grid level.</p>
Agent-based Controls	<p>This module refers to the traffic controllers with Control Agents for managing/controlling its operations, as described in <i>chapter 4, section 4.12</i>. Agent based Controls require a “Traffic Controller” which is different dependent on the type of traffic controller. The application provides the ability to host a “Control Agent” which provides Agent-Logic (in addition to default logic) and communicates with other Agents at Grid Level for advance decision making.</p>
Grid communication layer	<p>Includes various web services for interacting with other grid controllers and traffic controller devices. It provides all the connection management functions (Message Queuing and Session management), Security (Encryption and Authentication) and Notifications to other components etc.</p> <p>The communication layer also includes the “Agent Communication Layer” specifically designed to handle the Agent-based communication and interaction between different types of Agents (communication layer/interface described in <i>chapters 5 and 6</i>).</p>
Grid Knowledge base	<p>Comprises of databases for traffic Controllers/devices, Agents and Ontologies/Rules at grid level.</p>

Table 7.2: Grid modules and layers

Vehicle Controller (V_c): Application to manage and control in-vehicle systems/devices and provide communication with other vehicles and infrastructure grid controls. It also provides driver assistance features (Navigation/Route assistance, real-time feeds/alerts etc.) and can send its current location and status to the central server (for tracking purposes).

Remark: “Vehicle Controllers” and “Traffic Controllers” can be different types of system/devices and technologies. The ITS@CU platform proposed a design approach where a traffic Controller or Vehicle controller can provide the capability to host the Agents (as described in *chapter 4*). For the purposes of this research, various traffic controllers and vehicle controller applications were developed using a mobile application development framework (described later in this chapter, *section 7.4*) in order to simulate and analyse the design approach and the behaviour of Semantic Agent based Controls in the platform.

See “*Appendix H, Section 1*” for further details of the ITS@CU components and architecture description.

The *section 7.3* provides the technical description and implementation details of each of the platform’s component and modules.

7.2.3. SOA reference architecture of the platform

The overall architecture of the platform is based on SOA for the seamless integration between distributed traffic control systems and Agents from multiple domains using the Services mentioned in the previous section. The benefits and reasons for adopting SOA have been discussed in *chapter 2, section 2.5*.

Figure 7.3 shows the customised Reference Architecture which was designed as a framework to help and guide the overall SOA implementation of the platform. Reference Architecture is the architectural blueprint using well-defined SOA principles and best practices for describing all the services, their categories, layers, technical aspects, integration and underlying infrastructure.

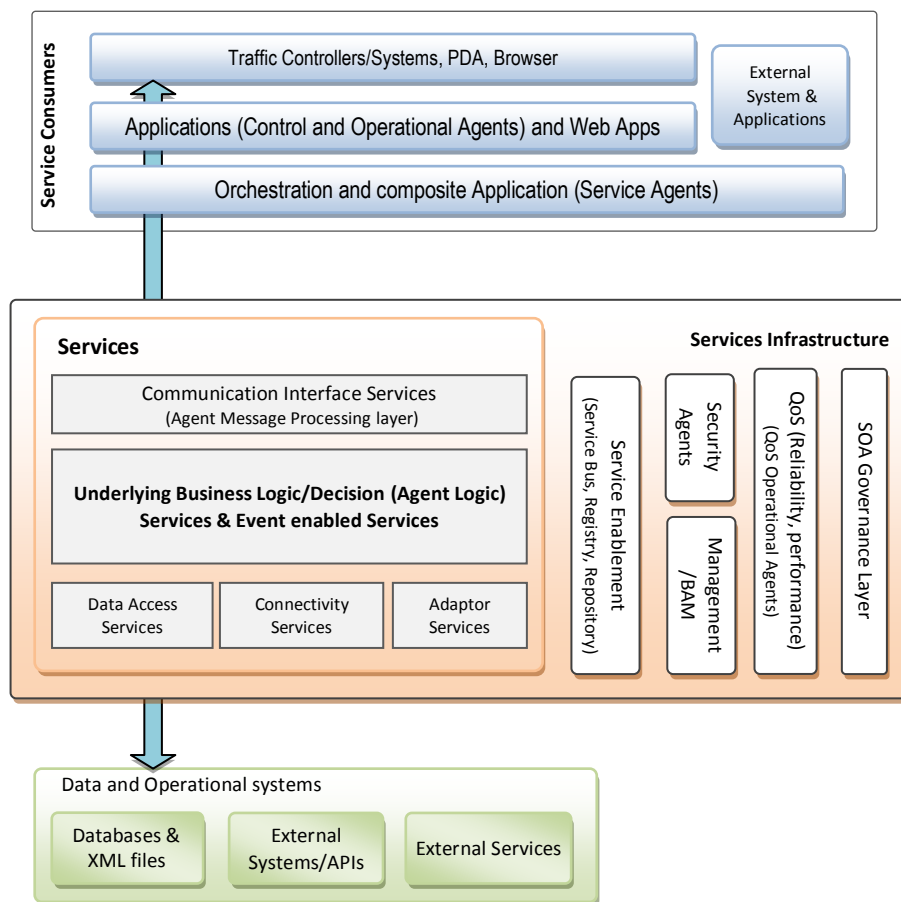


Figure 7.3: ITS@CU SOA Reference Architecture

As seen in *figure 7.3*, the platform's reference architecture has different layers comprising of different modules. The platform supports different types of service consumers (Controllers/PDA applications such as Grid Controller Application, Vehicle Controller simulation application or other platform management applications/utilities). These Controller applications support running relevant platform Agents hosted on the Controller devices. The Service Agents perform the composition of different

service requests to bundle into process flows through orchestration or choreography, which act together as a single or composite application. The composition of services in dynamic service flow is vital for Agent communication and decision making.

The core SOA layer comprises of the platform services and the supporting infrastructure modules. The services layer provides the main communication interface between different agents at both grid and central control system level. The “communication interface services” use message processing modules to interpret the incoming messages (as described in *Chapter 6, section 6.1* – communication flow) and uses the security Agent for authentication and access level. “Business logic/decision (Agent-Logic)” services are the main functional services providing the logic required for decision-making or to fulfil the requests made by service consumers (Agents). It also deals with internal and external events (System or Agents related). The Agent-logic services utilises “data access services” to access the knowledge base of the platform (i.e. Ontologies, rules and database). The “Connectivity services” and “Adapter services” facilitate the integration of the platform components (i.e. traffic controllers, grid controllers and external services).

The services in the platform are supported and managed by different infrastructure modules. The “Service Bus” uses different Operational Agents for Agent message routing, service translations, usage and orchestration features. Additionally, other modules use the Operational Agents (described in *chapter 4*) for instance QoS Operational Agents are used for service reliability/performance; the Service Registry Agent is used for dynamic service discovery; Agent Manager Operational Agents and Arbitrator Agents are used for management/Governance; and Security Agents provide authentication/encryption.

The consumer Agents dynamically discovers the services across the Service bus (using Service Registry Agents). Services publish their capabilities in the form of a WSDL description embedded with semantic/ontologies. The consumer Agents look for Tags (Meta key words) in the service descriptions to find, invoke and then choreograph into a composite service. Semantic web Services are interpreted based on their ontology description and the terms of use are based on the policies and service contracts (governance). These services interact with other services and underlying business components and data access components. The services can exist in isolation or as part of a composite service.

7.3. Components development and design description

This section describes the platform's components and its modules outlined in the previous section. It covers the technical implementation and design description including the technologies which were used to develop individual modules or applications.

7.3.1. Central Control system

The Central Control System comprises of various applications, databases and utilities responsible for managing the ITS@CU platform. It performs the following key functions:

- Traffic infrastructure Controllers, Agents and Grids management
- SOA enabled Services Layer for communication and coordination between all the components and Agents
- Intelligence Layer to oversee and arbitrate the decisions by Agents (on multi-grid level) regarding incident detection, response, self-organisation and traffic flow adaptation process
- Manage and provide ontologies and rules for Agent based communication
- Provides interface to external Services
- Real-time data analysis and central knowledge base
- Provide map data/GIS information and services (for route generation and Geocoding)
- Applications and utilities for map based visual monitoring and system administration

7.3.1.1. Infrastructure description

The Central Control application and utilities are designed to be hosted on multiple servers (running Windows Server 2008 R2). During the implementation of the research project they were hosted in a secure data centre (at T@lecom) and based on the following infrastructure as seen in *figure 7.4*. In this infrastructure approach there are three types of servers (clustered pair):

- Gateway Web Server
- Application Server
- Database Server

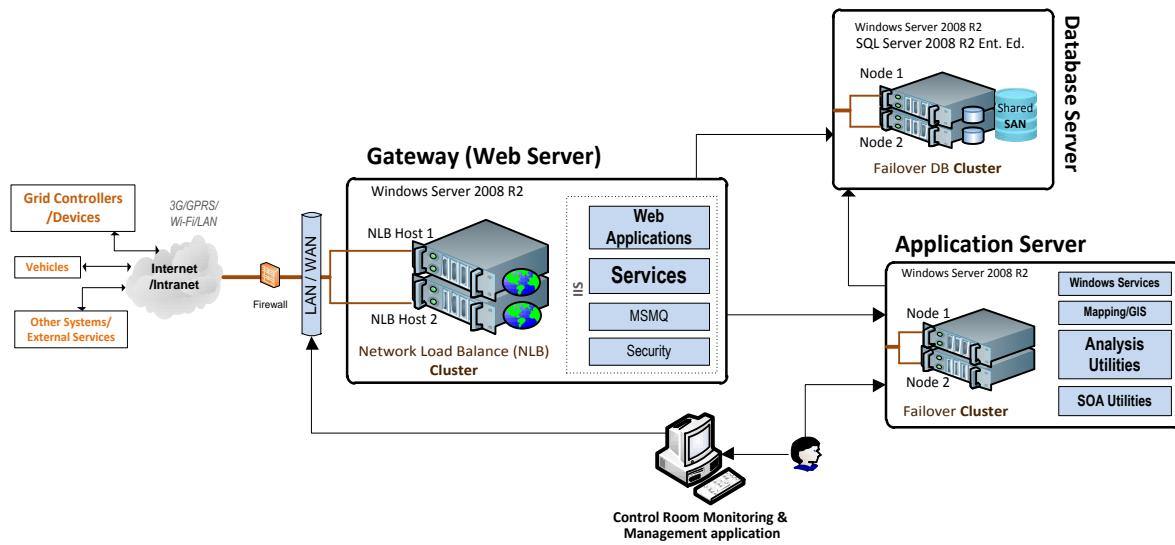


Figure 7.4: ITS@CU Central Control (Server Side) infrastructure

The **Web Server** provides an interface to other components/systems via .NET Web Services and Windows Communication Foundation (WCF) Services based on SOA principles and using a service bus. The Services and web applications are hosted on the Internet Information Services (IIS) and provide AES-128 based encryption for security purposes and GZIP based compression. Microsoft Message Queue (MSMQ) is used for message queuing and data reliability to avoid data loss during communication between Agents and Controller applications. The Web servers are configured as Network Load Balance (NLB) cluster for load sharing and also to provide a resilient interface to the end users (Grid Controllers/Devices – Agents).

The **Application Server** includes the core “Intelligence Layer” which consists of a set of windows services and desktop applications performing real-time traffic data analysis for detecting congestions and other problems. It also provides map data, GIS and routing information and interfaces with external systems (such as Microsoft Bing Map Services). It has a comprehensive map-based visual monitoring application and system management/administration application. Additionally, it has a simulation application allowing it to simulate the behaviour of Agent-based controls in various traffic situation such as congestion, road blocks etc. The Application Servers are configured as a Windows Server 2008 Failover Cluster for high availability and resiliency.

The **Database Server** hosts the “Knowledge base” which comprises of various databases for Grids Controllers/devices, Agents and Ontologies/Rules. It uses SQL Server 2008 R2 Enterprise Edition configured as a Failover Clustered application for high availability and resiliency. The data is stored on a shared Storage Area Network (SAN) device.

7.3.1.2. Services on the central control system

Communication services (interface/layer): This is the main communication layer for Agents and system component interactions hosted on the web server. It is the implementation of the Agent “Communication Interface” described in *chapters 5 and 6*. It is composed of various web services categorised into the following 3 types:

Security Service	<p>This service provides the following security functions:</p> <ul style="list-style-type: none"> Validates login credentials and access level of an Agent (and any other components/users in the platform) before granting it access to the required service in the platform Encrypt/Decrypt the AES-128 message contents
Notification Service	<p>This is a web service providing all of the event-driven notifications to relevant Agents. <i>It performs the following functions:</i></p> <ul style="list-style-type: none"> Broadcast messages to Agents (by groups) Notify Agents when updates are available (Agent-Logic update, Ontology/Rules update) Send stop, suspend or resume commands to agents (on behalf of Operational Agents) Send Alerts and Notification messages to traffic Controllers (Simulation mobile applications on PDA devices)
Agent communication Service	<p>This service provides all the key Multi-Agent communication between the Central Control and Grids level Agents. It provides a set of web methods/functions to facilitate SOA enabled communication between Agents of different types. <i>Some of the functions include:</i></p> <ul style="list-style-type: none"> Capability to interpret the message structure described in <i>Chapter 5, section 5.2</i> i.e. the commands/Instruction types used by Agents, and the semantic-contents interpretation using the ontologies and rules. Maintain connection sessions for messages (using Message-ID) and restore connectivity in the event of message failure or corruption Facilitate the conversations/message threads involving multiple messages by multiple Agents as part of the decision making process or composite service

	work flows. It uses MSMQ for queuing messages in an orderly fashion (based on priority basis) required for executing actions in a correct sequential manner.
--	--

Table 7.3: Communication Layer Services

The services in the communication interface/layer were developed in accordance with SOA principles as per the SOA reference architecture of the ITS@CU platform (discussed in *section 7.2.3*).

As seen in *figure 7.5*, the implemented web services allow SOAP, REST, HTTP Get and HTTP Post bindings and Port types for maximum compatibility and support for different types of service consumer.

Remark: All the Agent message instruction types (described in *chapter 5, section 5.2*) are implemented as web service operations with Semantic-Content, Ontologies and Rules embedded as XML data types. For example, the *AgtMsgRequest* operation in the port types seen in the following figure is the implementation of a “Request” message instruction.

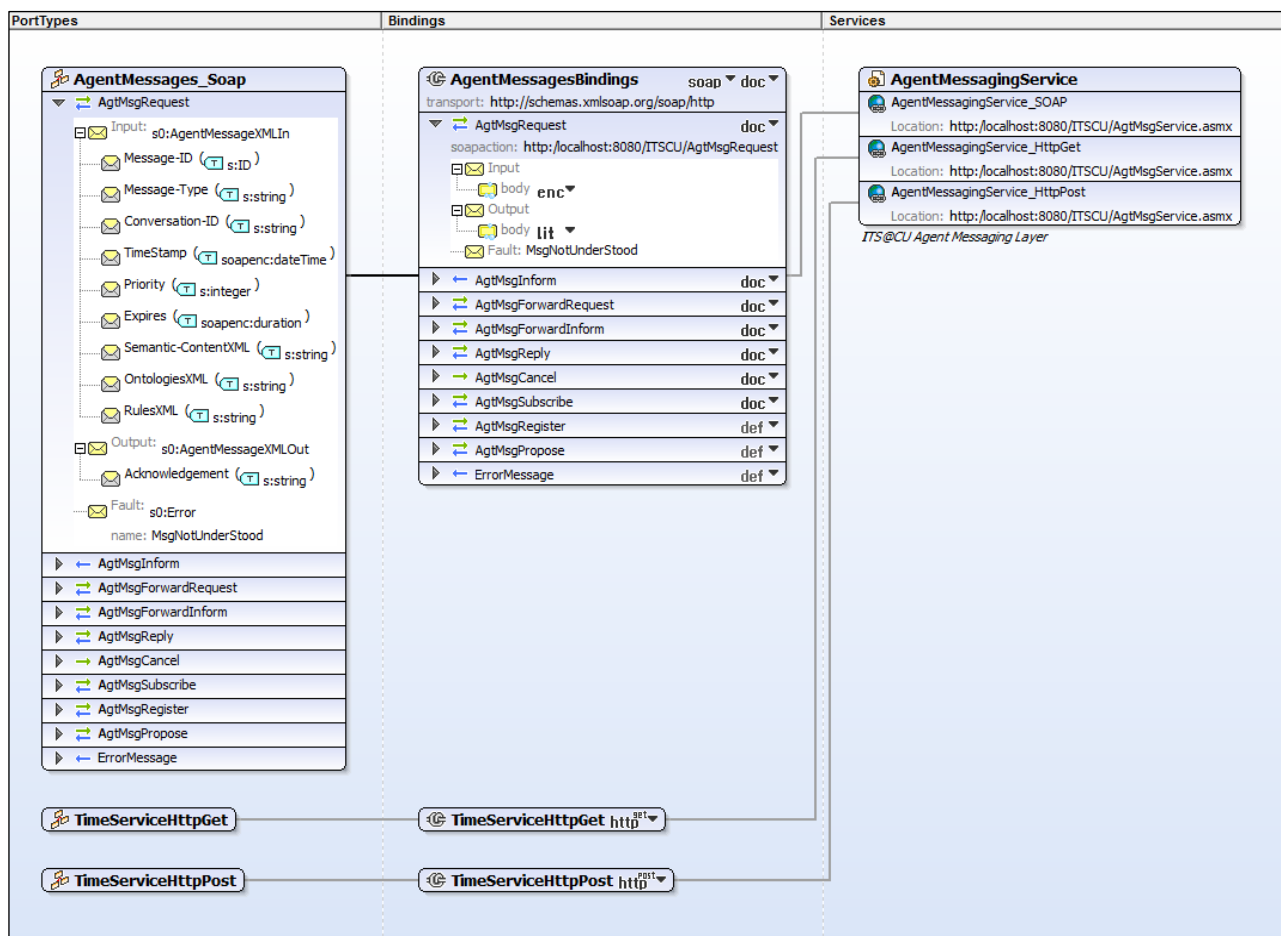


Figure 7.5: WSDL of the Agent Communication layer (shows Agent Message Instructions as web methods/Operations with semantic-content embedded as XML data type parameters)

The operations in the Agent communication layer service support Semantic Web Services. This provides flexibility in Agent based communication based on this research approach as traditional web service technology requires well defined service interfaces. Semantic web services address these limitations by augmenting the service descriptions with semantic metadata (meta-tags). The Semantic-Content (semantic layer) and Ontologies are therefore implemented as a core part of each operation in the Agent Communication interface/layer of the platform. The semantic web services also allow Agents to dynamically discover services and compose service flows benefitting on the adopted SOA approach as well.

Further details of the WSDL, class descriptions and code is provided in “*Appendix J*”

Platform Management Service: This service is used by the “ITS@CU System Management Application” and simulation utilities for the following management and administration related functionalities:

- Grid Management (Add/update or delete Grid Controller or change its settings)
- Agent management (Add/update or delete Agents in the platform)
- Traffic Controller management (Add/update or delete traffic controllers in the platform)
- Monitoring of traffic Controllers and Control Agents (used by Manager Operational Agents)
- Various other functions such as Database and logs clean-up

The Management Service is a combination of Windows Communication Foundation (WCF) and SOAP based web services hosted on the web server’s Internet Information Services (IIS). The Services were developed using .NET, C# and ASP.NET 2.0. The SOAP classes are modified using the SOAP extension mechanism allowing GZIP based compression and AES-128 encryption.

External Services: There are various external services in the platform (mentioned in *chapter 4, section 4.1.3*) such as Bing Map services and route info.

For the purpose of simulation and evaluation of the platform, various Web Service applications were also developed which include:

Weather Information	Simulation web service which provides a similar level of functionality as provided by the Met Office but with pre-configured data which can be altered in real-time to simulate different weather conditions
Route Information	Provides simulation route information data which can be configured to the street level

Table 7.4: Communication Layer Services

7.3.1.3. Applications in the central control system

The central control system consists of various applications and utilities performing different functions. The applications are of different types i.e. web application, desktop application or background windows service application. The following are some of the core applications:

ITS@CU System Management Application: This is a web based application for the managing, monitoring and administration of the platform. The main features include:

- Agents management (Add, delete, update, stop, suspend Agents)
- Grid and Traffic Controller Management
- Rules and Ontologies management (design and administration)
- System monitor and Update
- Bing Map based user interface for real time tracking of traffic Controllers and Control Agents
- Extensive reporting capabilities

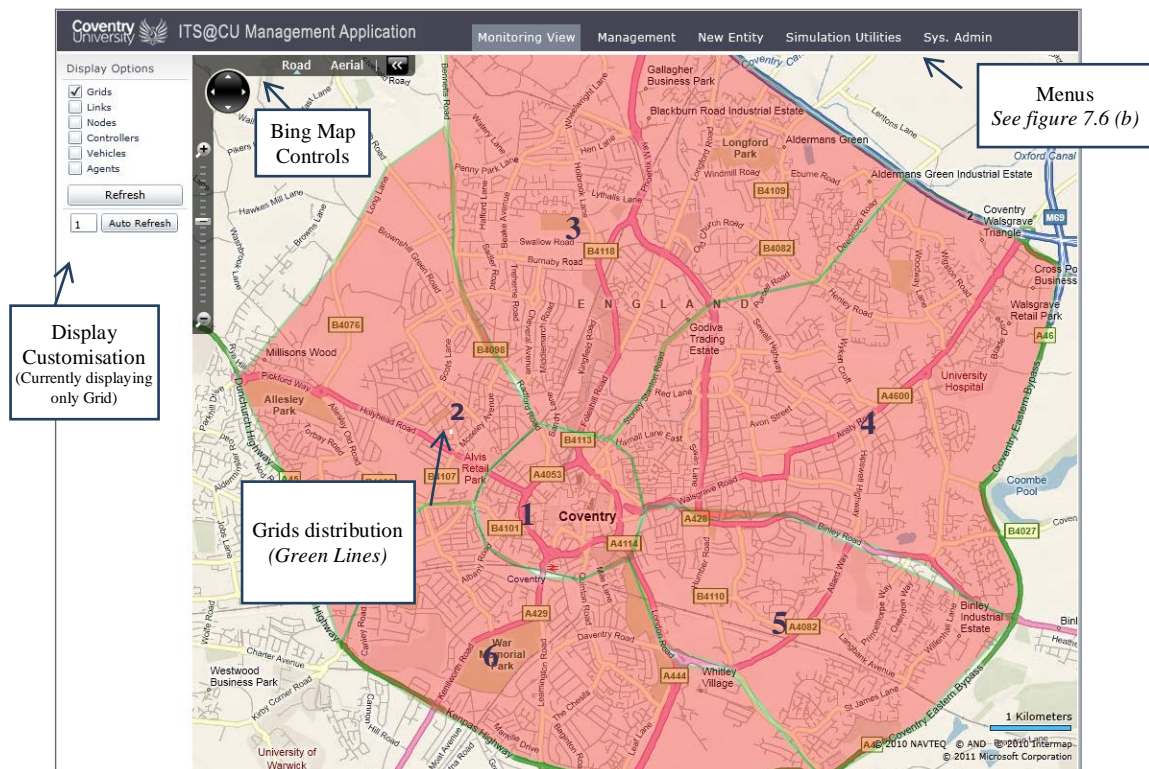


Figure 7.6 (a): ITS@CU Management Application (Default Monitoring View)

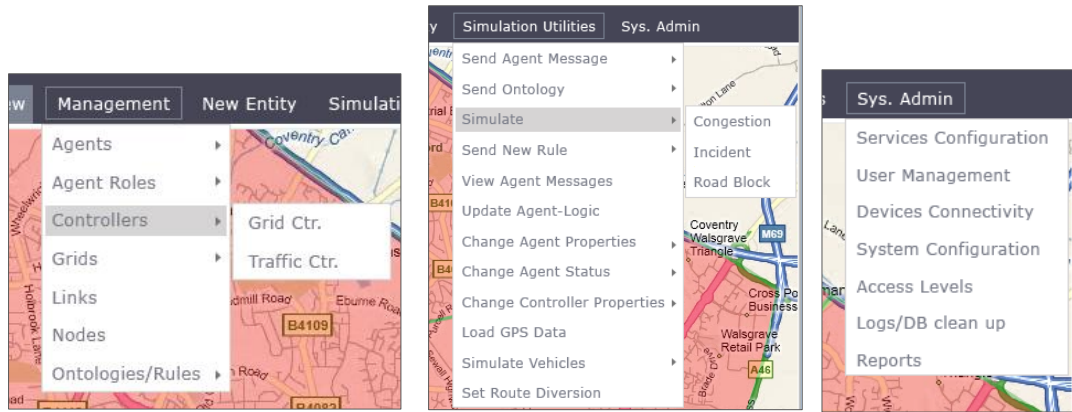


Figure 7.6 (b): ITS@CU Management Application
(Menu options for managing platform components, simulation options and administration features)

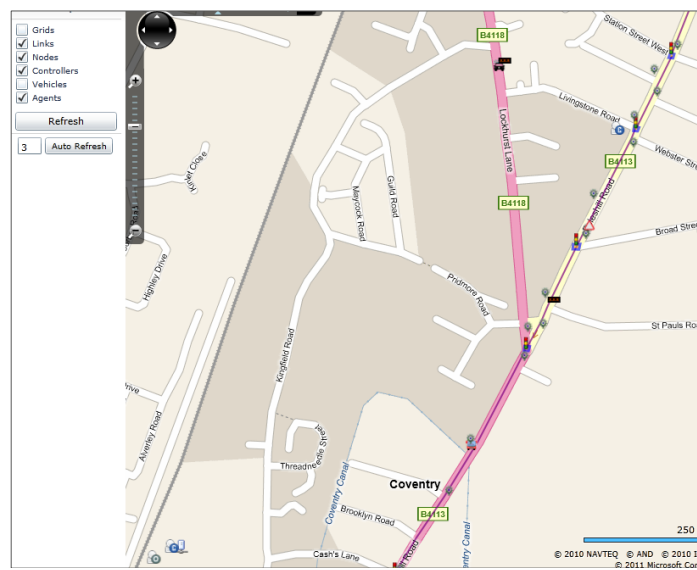


Figure 7.7: Monitoring view showing Traffic Controls icons on a route

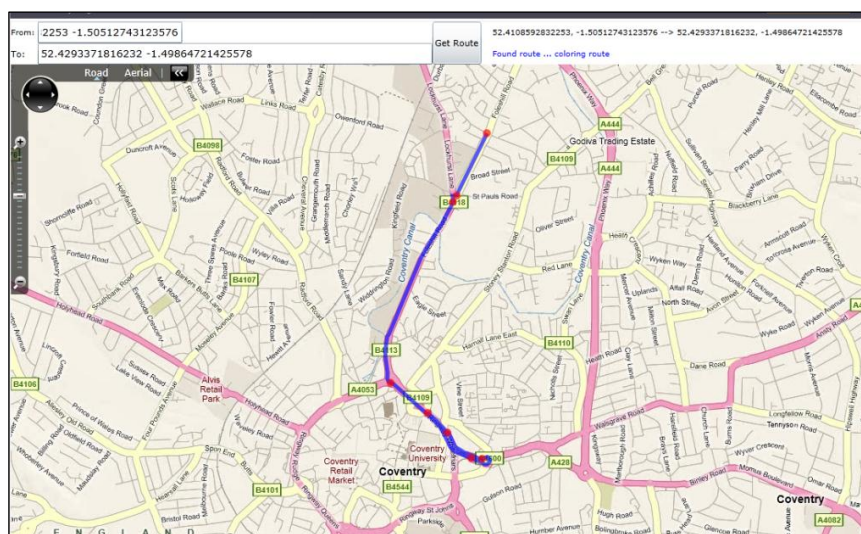


Figure 7.8: Automated Route diversion example
Agent uses Bing Maps Service to obtain directions between nodes in the platform and then analyse the shortest path

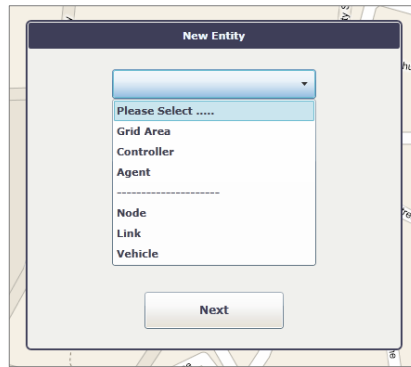


Figure 7.9 (a): Adding new entities (components)

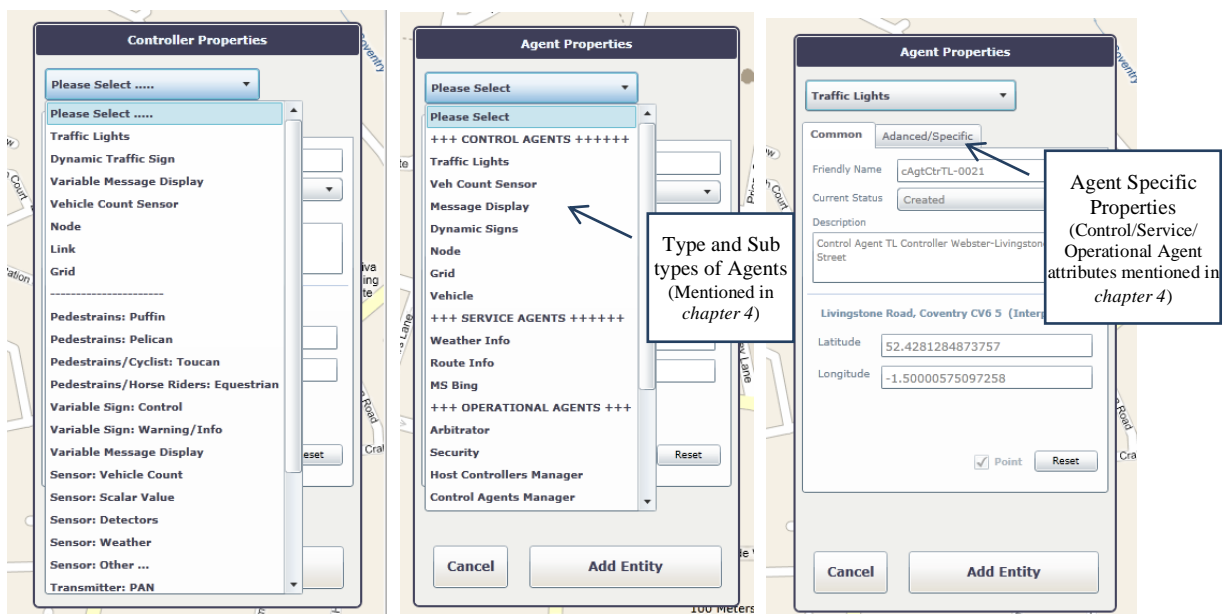


Figure 7.9 (b): Add new Controller (*left*), New Agent types (*Centre*), and New Agent Properties fields (*Right*)

See more screenshots of the system functionalities in “Appendix P, section 1 and 2”

This application was developed using the following technologies/tools:

- Microsoft Bing Maps API for extensive map integration and Route, Traffic and GIS services
- Microsoft Silverlight 4 technology which provided a rich in-browser UI experience and native support for Bing Maps.
- C# and Extensible Application Mark-up Language (XAML) as programming languages.
- AJAX and JavaScript for additional client side browser based application functionalities
- LINQ for database and XML data manipulation
- Windows Communication Foundation (WCF) for SOA enabled services

- SQL Server 2008 R2 as a core DBMS and Visual Studio 2008 as an IDE
- SSL/HTTPS and AES-128 based data encryption for security

Platform Intelligence Layer Application: This is the core application and key to all Agent-based interactions and decision making processes in the platform. It is a Windows Service Application which runs as a background process on the application server. The main features of the application include:

- Access central knowledge base on the database Server
- Real-time data analysis of Agents and traffic Controllers
- Control and arbitrate the decisions by Agents at the grid-level regarding incident detection, response, self-organisation and traffic flow adaptation process (currently limited to pre-configured routes, ontologies and rules only)
- Additionally, it hosts Operational Agents (Central Control Level) and Service Agents for providing access to external service.

The application was developed using the following technologies/tools:

- .NET Framework 3.5
- C# as a programming language
- LINQ for database and XML data manipulation
- SQL Server 2008 R2 as a core DBMS

Simulation utilities/applications: Additionally, various other utilities were developed to simulate the behaviour of Agent-based controls in various traffic situations such as congestion, road blocks etc.



Figure 7.10: Generating an Agent Message (behaviour testing/simulation)

See more screenshots of the simulation utility in “*Appendix P, section 3*”

7.3.1.4. Central Database/Knowledge base of the platform

ITS@CU has a main database and various supporting backend databases to form the knowledge base of the platform. *Figure 7.11* shows the structure of the main database of the ITS@CU platform reflecting the road network model elements (described in *chapter 3*), Agent types, roles and organisation structure (described in *chapter 4*), and the Communication and the Ontology design (described in *chapter 6*).

The main database stores/provides the following key data:

- Agents and their organisational relationships (types, roles, attributes, access level and team/agency)
- Agents code logic/functions (defined in XML)
- Agent messages and conversation threads
- Rules and Ontologies for Agents communication (defined in XML)
- Platform Controllers/Systems (Grids, nodes, links, traffic controllers etc.)
- External Services and their descriptions in XML for Service Agents

Other supporting databases store/provide:

- Traffic Controls status (for simulation purpose)
- Vehicles and their current location for traffic analysis (optional or for simulation)
- Administration Users and their credentials

The databases are hosted on the SQL Server 2008 R2 Enterprise Edition in a clustered configuration. The databases are normalised to third normal form (3NF) to ensure efficient database queries and transactions. The main database also uses Stored Procedures for a high level of flexibility and optimisation, so application can access pre-compiled Stored Procedures which can be updated outside of application Logic. This was also important for updating the runtime Agents' behaviour without changing any code in the platform applications. Other than Stored Procedures the applications and Services also access the database using Language Integrated Query (LINQ) which provides native C# database access functions.

The main database also uses Spatial Data (also known as geospatial data or geographic information) which is a new GIS feature of SQL Server 2008 to identify the geographic location and features such as natural or constructed features (e.g. roads, buildings, landmarks, rivers etc.). Spatial data is stored as coordinates and is compatible with most mapping solutions such Bing, MapPoint and Google Maps.

As seen in the *figure 7.11*, the ontologies and rules are stored as XML data type (in database table Ontologies column **OntologyXMLDesc**, and table Rules column **RulesXMLDesc**). SQL Server provides native support for XML document storage and parsing capabilities. It also allows Transact-SQL (T-SQL) queries within the nodes and elements of the XML column contents which reduced parsing and processing at the application level.

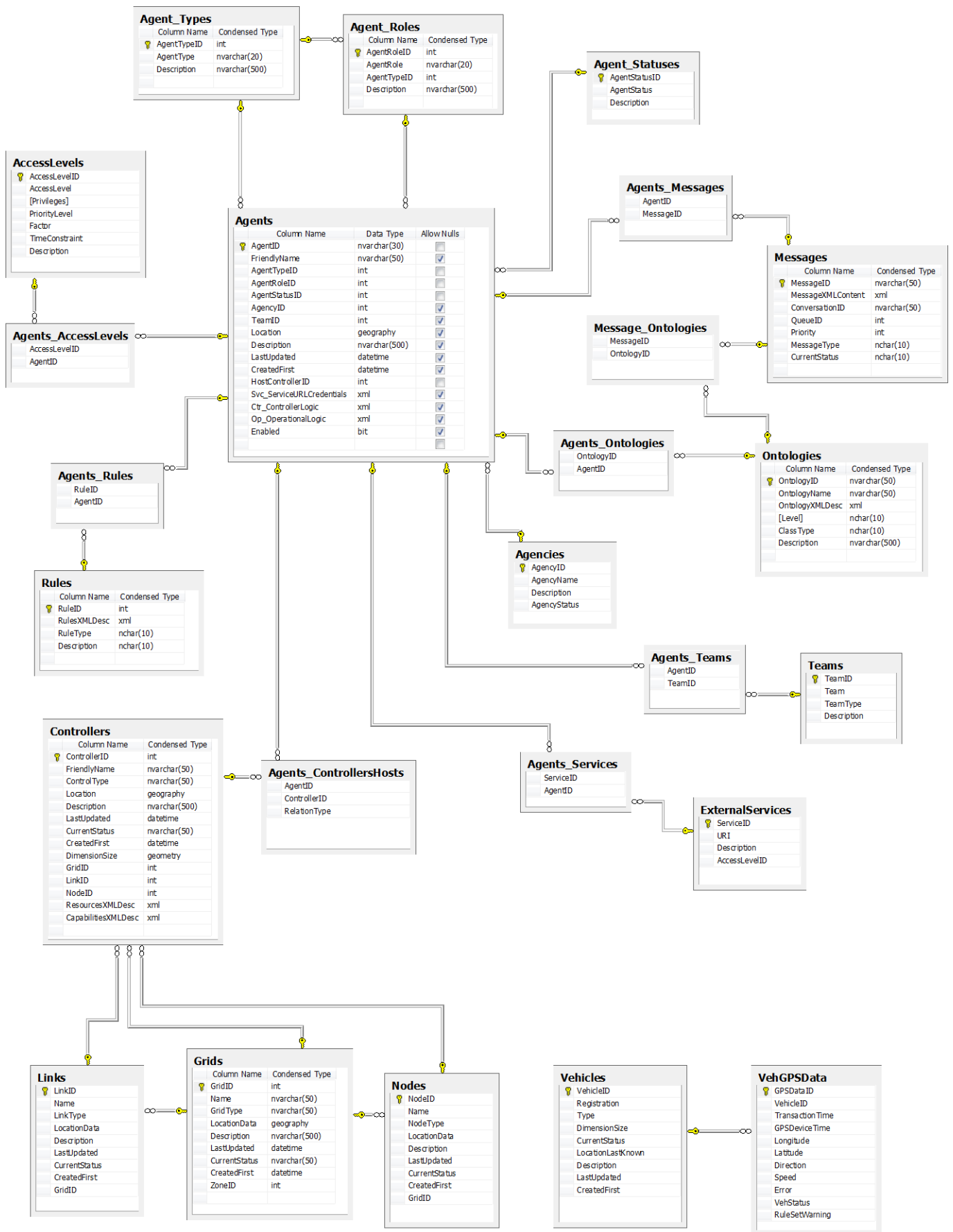


Figure 7.11: ITS@CU Main Database Structure

7.3.2. Traffic infrastructure/Grid Controls

The ITS@CU platform requires that the traffic infrastructure be divided into smaller grids (zones/areas) for better manageability of the traffic network devices and to enable the Agent based Controls to function efficiently. Each grid has a “Grid Controller” application for controlling and managing the Grid and the associated traffic controllers. It provides the ability to host Agents and is responsible for the decision making/arbitration capabilities at grid level.

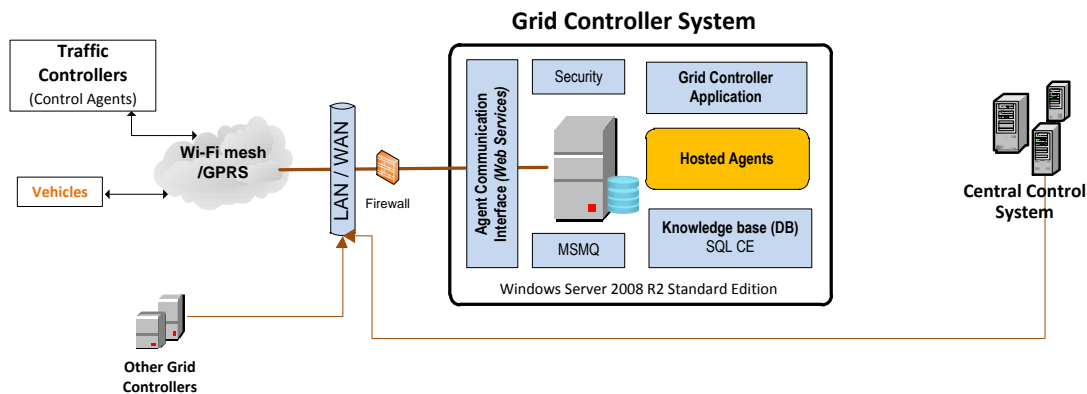


Figure 7.12: Grid Controller system Architecture

This “Grid Controller” application is using a server with the Windows Server 2008 standard edition operation system. A Grid Controller application can work on normal desktop or even Windows CE mobile device however for resiliency and reliability windows Server 2008, ideally clustered, is preferred.

7.3.2.1. Grid Controller Application

This is a .NET based windows forms application developed in C#. The main features of the application include:

- Control and arbitrates the decisions by Agents on “grid and controller level” regarding incident detection, response, self-organisation and traffic flow adaptation process
- Real-time data analysis of local traffic Controllers and Agents
- Grid level Agents management (Add, delete, update, stop, suspend Agents)
- Local Traffic Controller Management
- Local Rules and Ontologies basic management
- System monitor and basic administration

- Agent Communication interface/Layer for Agent Communication

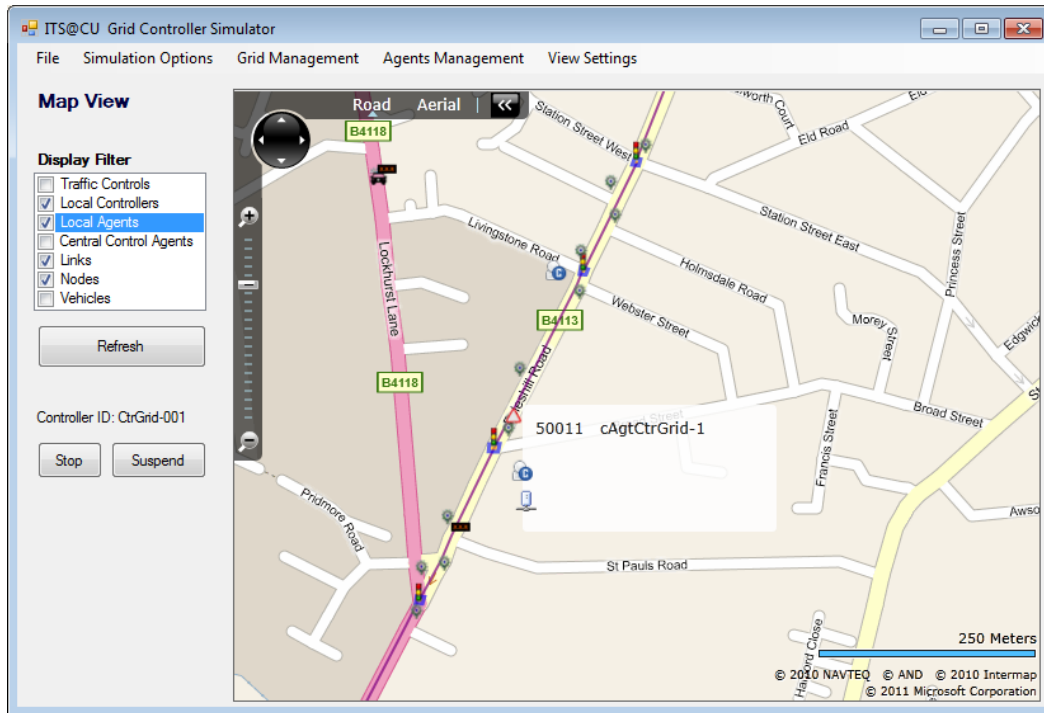


Figure 7.13: Grid Controller Application
(Default Map view showing the traffic controls on a route)

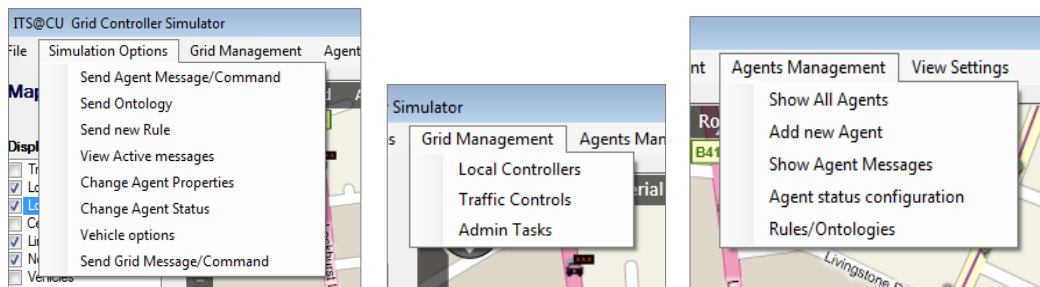


Figure 7.14: Grid Controller Application
(Menu options for managing local traffic controllers/Controls, Grid Level Agents and various simulation features)

7.3.2.2. Agent Communication Interface/Layer

A Grid controller system also provides communication capabilities similar to the Agent Communication Layer of the Central Control System (described in *section 7.3.1.2*), however it includes only a limited set of operations which are relevant to Grid level Agents. The agent communication interface is part of the Grid Controller Application which supports multi-channel communication (Web Services and Socket listeners) with SSL encryption, GZIP compression and MSMQ.

Communication technologies: During the design of the platform various communication technologies were investigated (see “*Appendix G*”) in order to create a communication layer between the various components of the platform such as communication between Controller to Grid Controller, Grid Controller to Central Control, Vehicle to Grid, vehicle to Central Control and Vehicle-to-vehicle. ITS@CU is not dependent on any specific communication protocol however HTTP based communication is preferred (and used in the platform implementation) due to its SOA support. Similarly, various communication channels are supported such as Bluetooth, IrDA for MANET (mobile ad-hoc network) and GPRS/3G/Wi-Fi for WAN connectivity.

7.3.2.3. Grid Database/Knowledge base

In addition to the central database, each grid controller has its local database holding information about the nodes, links, Agents and local traffic controllers. Similarly to the main central database, the grid database also reflects the road network model elements (described in *chapter 3*), Agent types, roles and organisation structure (described in *chapter 4*), and the Communication and the Ontology design (described in *chapter 6*). However the Grid database does not include the full platform data i.e. it only stores and provides data relevant to the grid such as:

- Control Agents and grid level Operational Agents and their structural details
- Control Agents code logic/functions (defined in XML)
- Messages between Agents (inter grid only)
- Rules and Ontologies for Agent communication
- Traffic Controllers/host details
- Traffic Controls operational status data

Unlike the main central database, the grid database is light-weight, and can be hosted on either SQL Server 2008 or compact edition. In this research, SQL Server 3.5 Compact Edition was used as the Grid controller application is designed to be hosted on devices with limited available processing.

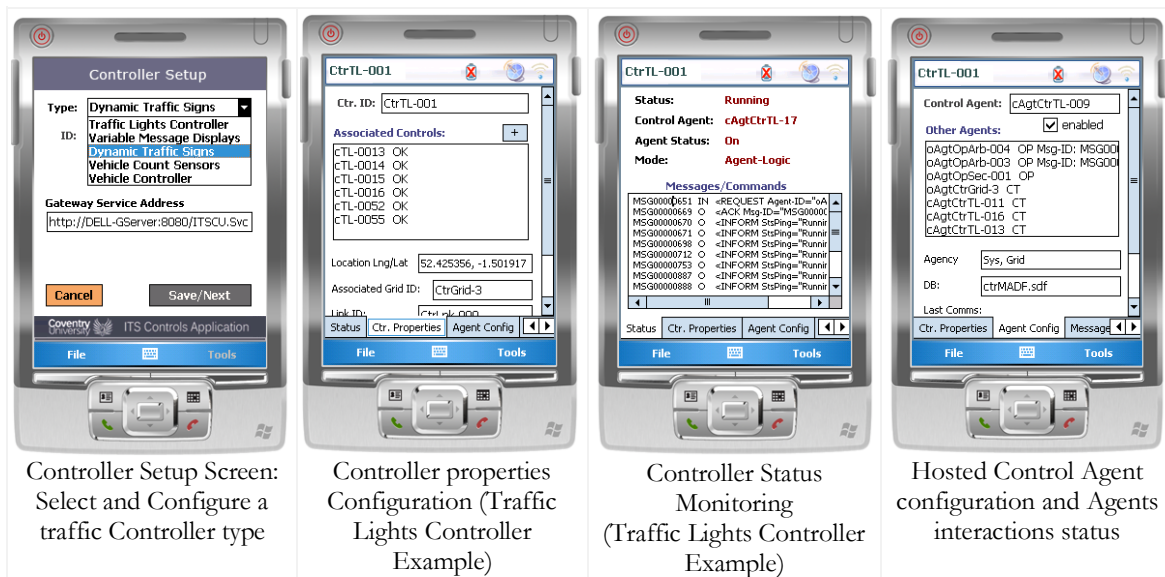
Ontologies and rules are stored as XML in table columns and Agents access/manipulate the grid database (via the Grid Controller) using Language Integrated Query (LINQ).

7.3.2.4. Traffic Controllers applications

As mentioned in *chapter 2* and *4*, modern urban traffic management systems includes a set/group of specific traffic control devices (e.g. traffic lights or sensors) controlled by a local “Controller” responsible for their operations. The traffic control devices just perform specialised tasks. These Controllers are interconnected and monitored by central or regional control centres. For example, in the Coventry City traffic management system, all the traffic light and inductive loop detectors/sensors are controlled by their local Controllers (called out-station transmission unit (OTU)) which are interlinked (using built-in modem) and integrated with SCOOT system for overall monitoring and calibration of traffic flow and controllers (see “Appendix L” for details).

Traffic Controllers and Controls can be different types of systems/devices and technologies therefore it was not feasible to use real traffic controls in this research. In order to analyse the Agent-based Controls design approach (discussed in *chapter 4* and specifically in *section 4.1.2*) and to simulate their behaviour in the platform, various traffic Controller and Control applications were developed (using the Mobile Application Development Framework (MADF) described later in this chapter, *section 7.4*). MADF supports both “Agent-Logic” and “Default-Logic” modes of the Controllers.

The following are some of the Applications developed for simulating the behaviour of the Controllers:



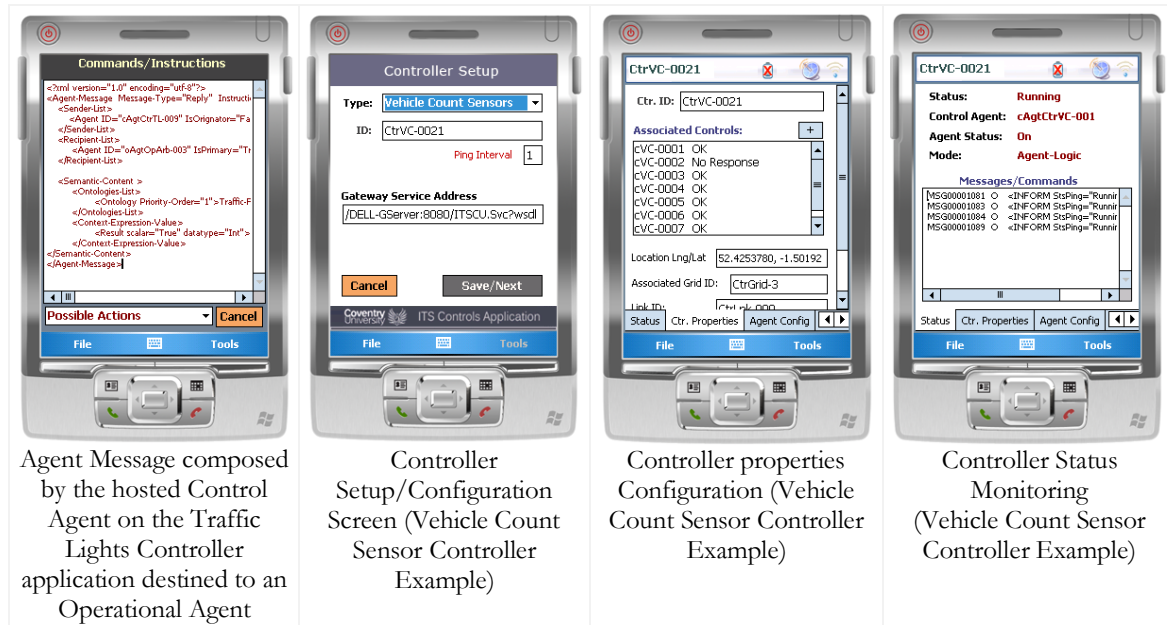


Figure 7.15: Controller Application (for Simulation)

See more screenshots of Controller Applications in “Appendix P, section 4”

The following are screenshots of some applications simulating the behaviour of local traffic controls associated with the above Controllers. See more screenshots of Traffic Control Applications in “Appendix P, section 5”

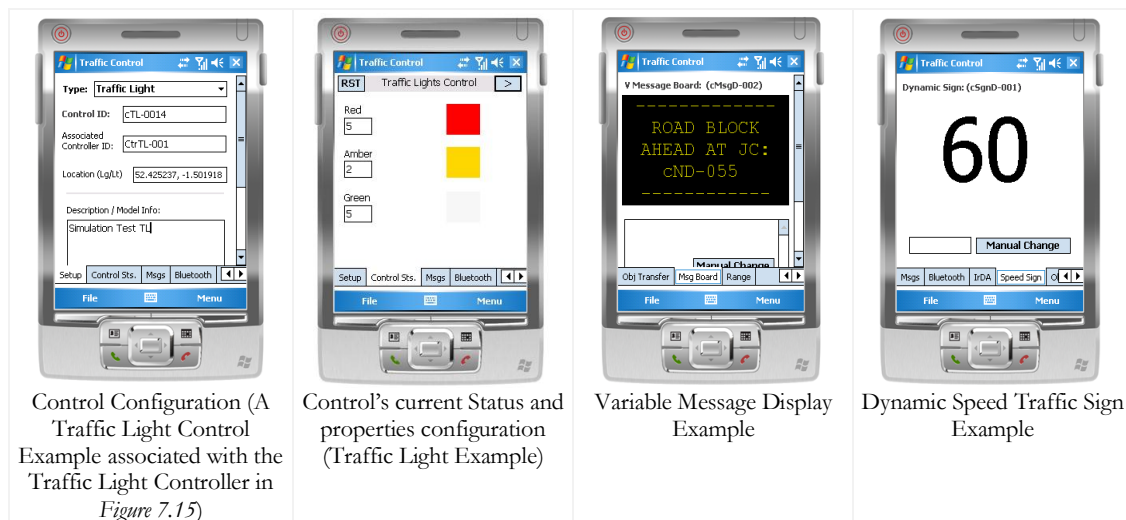


Figure 7.16: Local Traffic Controls Simulation Application

7.3.3. Vehicle Control System

Vehicles are controlled by a “vehicle controller” application which manages and controls in-vehicle systems/devices and provides communication with other vehicles and infrastructure grid controls. It also provides driver assistance features (Navigation/Route assistance, real-time feeds/alerts etc.) and sends the current location and status to the central server (for tracking purposes).

“Vehicle Controllers” can be different types of systems/devices and technologies (depending on the manufacturers). For the purpose of the research, a vehicle controller application was developed using a mobile application development framework (described later in this chapter, *section 7.4*) in order to simulate vehicles in the platform. The Vehicle controller is an on-board mobile/PDA application (based on Windows Mobile 5/6.5) and local database (SQL Compact Edition 3.5). The application is fully integrated with navigation software such as TomTom6/7 and Co-Pilot.

The vehicle controller application supports multiple wireless communications channels (Wi-Fi, 3G/GPRS) for Vehicle to Grid and Vehicle to Central Control Services communication. The communication is mainly via web services however the calls are compressed to avoid slow transmission especially in poor GPRS coverage areas. It also supports Bluetooth and IrDA to form ad-hoc networks required for Vehicle to Vehicle, Intra-Vehicle and Vehicle to roadside controls communication. Local traffic information can be pushed to vehicle controls likely to be affected by the observed or predicted congestions.

See “*Appendix G, Section 2*” for Vehicle to Vehicle based ad-hoc network application details.

The following are some screenshots of the vehicle Controller application showing various functions:

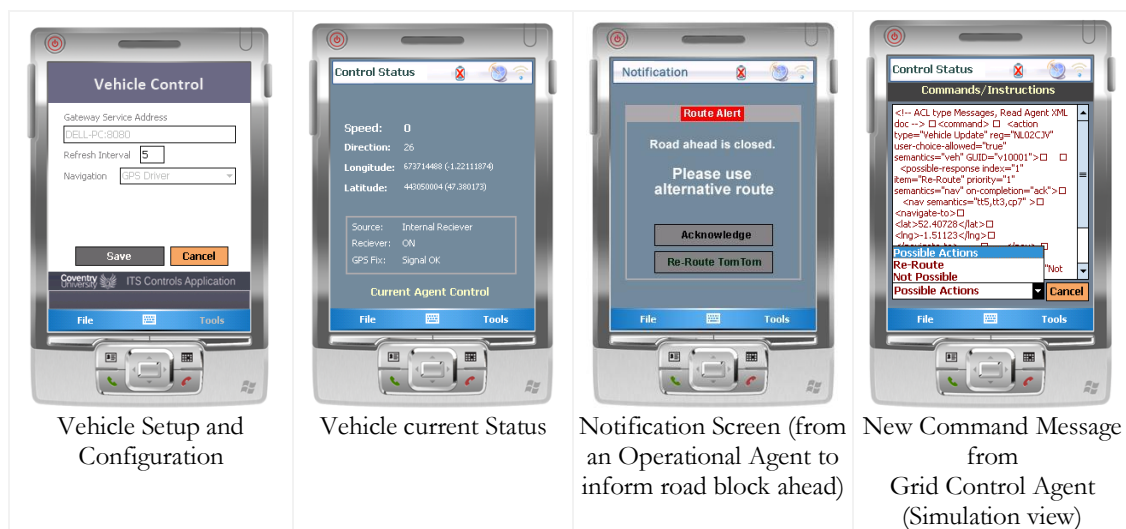


Figure 7.17: Local Traffic Controls Simulation Application

See further screenshots of Vehicle Control Application in “*Appendix P, section 6*”.

Figure 7.18 illustrates the Vehicle Control system architecture, sub-components and its interaction with other components.

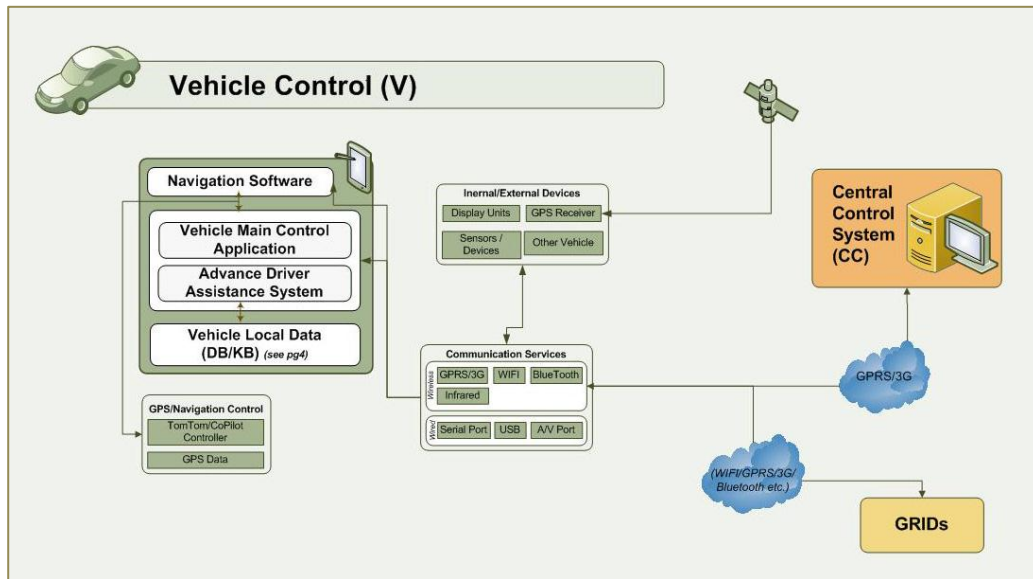


Figure 7.18: Vehicle Control Architecture

See “Appendix H, Section 2” for further design level details of the vehicle control component/modules

7.4. Mobile Application Development Framework (MADF)

MADF was an important element of this research. It was specifically developed to support the ITS@CU platform and the novel communication approach based on Semantic Agent-based controls described in *chapters 5 and 6*. It allows rapid application development for simulating traffic Controllers and Control Agents. It supports and provides modes for all of the ITS-based components and Agent controls (sensors, traffic signals, dynamic signs, in-vehicle applications, grid controllers etc.). It also has multi-channel wireless capabilities (3G/GPRS, Wi-Fi, Bluetooth) to enable communication between the various components and Agent-based Controls.

The focus of this research and its requirements was to utilise mobile devices such as PDAs for simulating ITS controls and vehicle controls. A PDA can provide applications for navigation, driver information system, wireless vehicle-vehicle communication (using Bluetooth, infrared and Wi-Fi), and vehicle-to-grid and vehicle-to-control centre simulation and communication (using Wi-Fi and GPRS/3G/HSPA).

The author spent considerable time (about a year) in developing the MADF using the Microsoft .NET Compact Framework, C#, XML, Web Services, SQL Server CE, Windows Mobile 5/6 SDK, and Microsoft Visual Studio 08. The reasons for adopting .NET Compact Framework, C# and Visual Studio are discussed in detail in “*Appendix F, section – tools and technologies*”.

MADF enables the development of any type of mobile application for the ITS platform in a very simplified way. The framework provides various Project Templates in Microsoft Visual Studio 2008, and an extensive set of C# libraries (dll files). It also includes Visual Studio based custom controls such as GPS modules or traffic light modules etc., all available as tools allowing for a simple drag and drop development environment.

Some of the core features of MADF include:

- Traffic controllers are configurable in both “Agent-Logic” and “Default-Logic” modes which is important for testing the Agent-based Controls approach and compare with the traditional fixed control behaviour of the Controllers.
- Agent Communication Layer (described in *chapter 5*) support with custom XML parsing module for Agents Messages, Ontologies and Rules interpretation
- Multi-Agent hosting support using multi-threaded objects
- Wireless Communication layer: Bluetooth and Infrared based Personal Area Network (for small range based ad-hoc networking and communication between Vehicle-2-Vehicle and Vehicle-2-Infrastrucutre

- Support for the “Communication Interface/Layer Services” mentioned in *section 7.3.1.2*: for communication between the Control Centre server and vehicles and infrastructure devices (over 3G/GPRS and Wi-Fi)
- Vehicle controller mode (for different types of vehicles i.e. normal or emergency),
- Traffic infrastructure device control modes (such as traffic lights, variable message displays, and signs)
- SQL Server CE Database support (for storing local data)
- GPS data collection and analysis using built-in receivers
- Navigation application control (using TomTom 5/6 and CoPilot 7 API)

MADF was used for all of the mobile applications (Traffic Controllers/Controls and Vehicle Controller) in this research for simulation.

See “*Appendix P, Sections 4, 5 and 6*” for applications developed using MADF.

MADF Architecture and Design

The architecture of the MADF follows Model View Presenter (MVP) approach and all the components and libraries are layered, segregating and hiding the core functionalities from the main code.

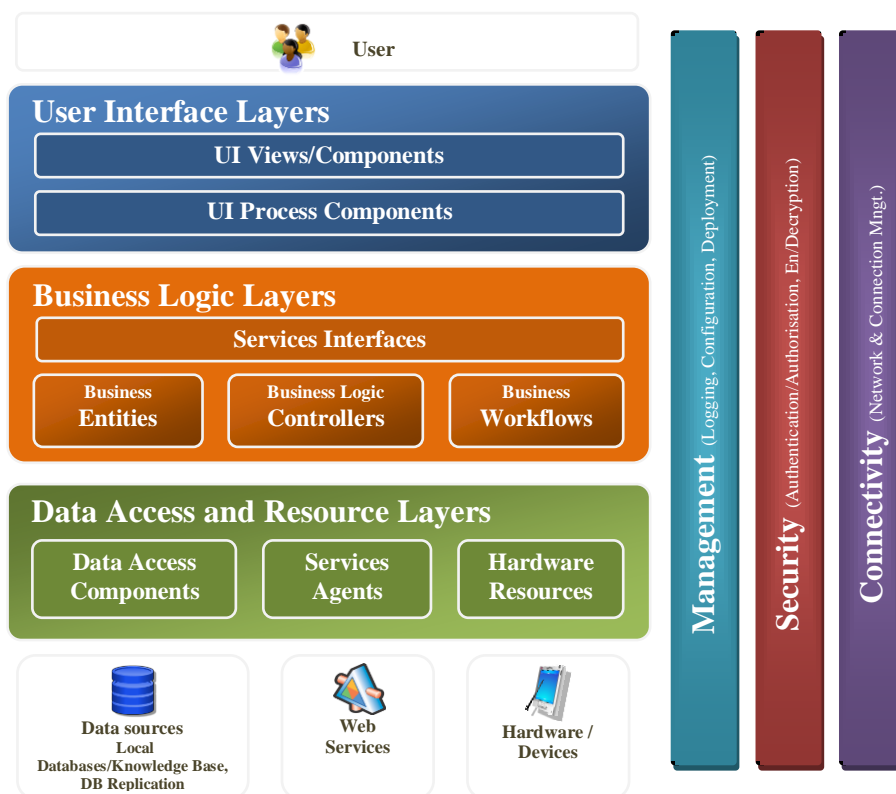


Figure 7.19: MADF Application Architecture

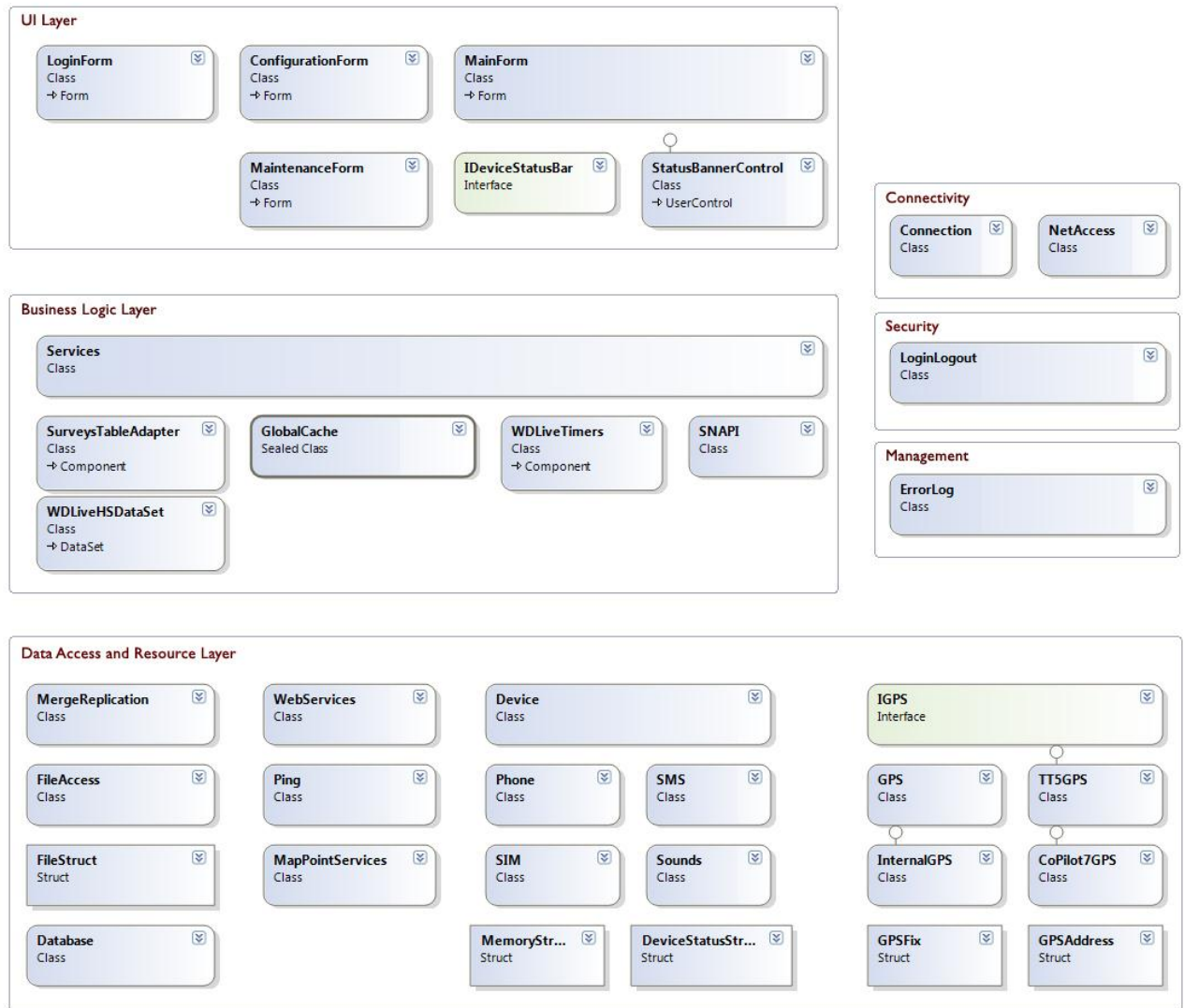


Figure 7.20: MADF classes & layers

Remark: The MADF is T@lecom’s Intellectual Property Right (IPR) as per the KTP contractual terms. Only high-level and relevant description is presented in *figure 7.20* and only commercially insensitive and approved level of details about the MADF classes and methods are provided in “*Appendix P*” and the source code/instruction included in “*Appendix J, CD*”.

7.5. Conclusion

This chapter presented the technical design and development description of the ITS@CU platform (and associated components/utilities) which was specifically developed to implement and evaluate the Semantic Agent-based Controls approach in this research. The platform enables the multi-agent communication layer approach (presented in chapter 5 and 6) using SOA principles as an underlying implementation and integration architecture. The platform was developed using commercially available and state of the art proven technologies at the time of development.

The later part of the chapter presented the MADF, a supporting mobile applications framework developed for ITS@CU. It provides rapid mobile/PDA application development capabilities for developing different traffic Controller and Control applications in order to simulate the behaviour of traffic control devices and vehicles. The MADF was important for evaluating the research approaches without focusing on the underlying technology of the actual traffic controllers, and for its support for the Agent communication layer and Control Agent hosting capability.

Chapter 8

Evaluation & Analysis

The previous chapter presented the technical design and development description of the ITS@CU platform.

This chapter discusses and evaluates the overall research approach of using the novel concepts of Semantic Agent-based Controls and their communication and co-ordination in large scale SOA based ITS systems. It presents an analysis of the ITS@CU platform by simulating different test cases. The outcome of the simulations are analysed to demonstrate the advantages of the proposed implementation.

The chapter is divided into the following sections:

- The first section describes the simulation system setup, geographical area studied, the test data and Agents configuration
- The second section presents an analysis of the different traffic scenarios and test cases in order to discuss how they achieve the research objectives
- The third section discusses the limitations and future potential of the overall research approach
- The final section of the chapter concludes the overall evaluation findings

8.1. Evaluation approach and simulation setup

8.1.1. Study area overview

During the research, the author liaised with UTMC Control Room, a department of Coventry City Council. They provided real traffic data for different routes at different times/days (See “*Appendix L*” for a report outlining the collected data and traffic trends).

This liaison with Coventry City Council assisted this research with the following:

- identifying the key routes and junctions for evaluating the platform
- providing practical traffic control expertise and evaluating and refining the research approach
- providing access to historical traffic data and traffic flow trends

Coventry City was selected as the main geographical area of study due to its proximity to the research establishment, the nature of the current system in place within the city and the levels of access the city council were willing to provide to personnel and traffic data.

Using the Road Network Model described in *chapter 3* and for the purposes of evaluation, the city has been divided into 6 Grids (as shown in the *figure 8.1* below). The city centre has been allocated a smaller Grid as there is a greater concentration of traffic controls and other elements when compared to other Grids.

The traffic elements (Nodes, links Controllers, Controls) in the Grids were created/placed on their current location as per the data provided by the Coventry City Council. Additional Traffic Controls/Elements were added to the system in different places for testing and evaluation purposes and to assess possible improvements to the current traffic network.

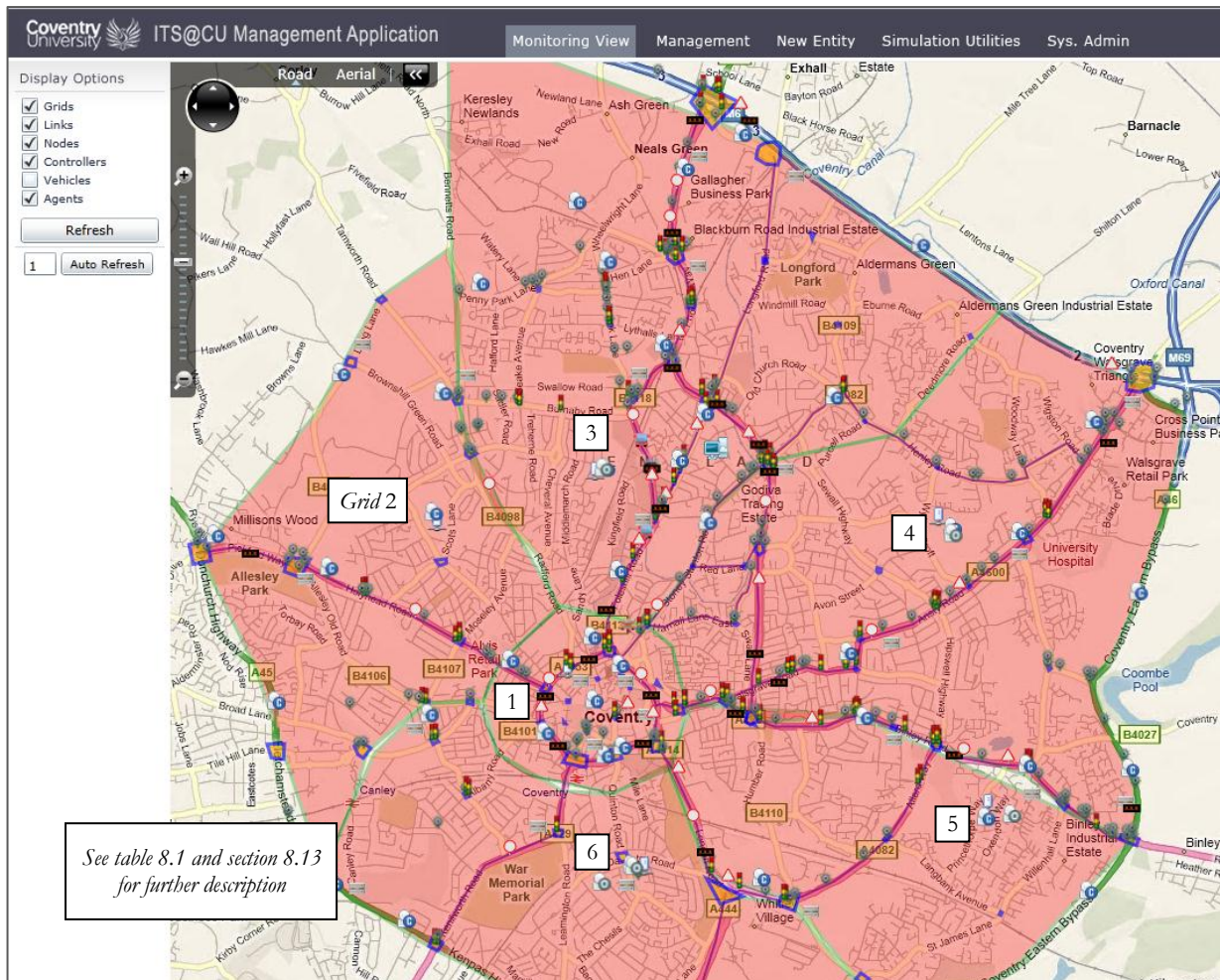


Figure 8.1: Study area (Coventry City Centre) with traffic Controls, Elements and Agents

8.1.2. Simulation setup overview

ITS@CU was evaluated in several phases and the simulation studies were performed using the hardware infrastructure/configuration and applications detailed in “Chapter 7”. Several test servers running Windows Server 2008 R2 were configured at T@lecom’s data centre for the Central Control System. The Grid Controllers were hosted on Virtual Server Machines (using VMWare virtualisation software).

The Traffic Controllers, Vehicle Controllers and the local traffic Controls were set up using the MADF simulation mobile application (described in Chapter 7, section 7.3.2.4 and 7.3.3). The mobile simulation applications were installed on Windows Mobile 6.5 based PDA devices with wireless connectivity (i.e. Wi-Fi, 3G/GPRS) for Controllers to Grids/Central Control communication and Bluetooth for ad-hoc networking simulating vehicle to vehicle and vehicle to infrastructure communication.

A Virtual environment on a smaller scale (using one Server, laptops and a few PDA devices interconnected together by Wi-Fi) was set up at CTAC test lab at Coventry University. It was also used at different stages of the research for evaluation.



Remark: Due to the large number of traffic Controls and Controllers required, and to eliminate the hardware differences and intermittent communication factors, PDA emulation software (Windows Mobile 6.5 Phone Emulator in Visual Studio 2008) was used, enabling the simulation of multiple Controls and Controllers on a PC/Server. The Controllers were configured to support both “Agent-Logic” mode and “Default-Logic” mode (supported by the Controller simulation PDA application described in *chapter 7*) in order to test the Agent-based Controls approach and where applicable compare with the traditional fixed control behaviour of the Controllers (as described in *chapter 4*).






The Route and Weather web services (used as examples of external services for the system) were hosted on the Central Control Gateway Server.






The Agents, Controllers and Controls were set up using the platform Management Application described in *Chapter 7, section 7.3.1.3*.

8.1.3. Platform and Agents configuration

The following entities (traffic elements, Controllers and Controls) were configured for the simulation studies discussed in the next section of the chapter.

Platform Entity	Notation	Image	#	Description
Central Control System	<i>CtrlCc-1</i>		1	<p>The Central Control System was set up on several test servers according to the infrastructure design described in <i>Chapter 7, section 7.3.1.1</i>.</p> <p>For the purpose of the simulation, the location of the Central Control System (as seen in the <i>figure 8.1</i>) is set as the current location of the Traffic Control Centre, UTMCC Control Room, Coventry City Council.</p>
Grid			6	The traffic network area was divided into 6 grids, based on the Road Network Model and on recommendations by Coventry Council.

Grid Controller	<i>CtrlGrid-n</i>		6	Each grid had a Grid Controller Application hosted on a Virtual Server/Machine (VM). 6 VM's were set up representing the grid control infrastructure design similar to the one described in <i>Chapter 7, section 7.3.2</i> .
Node	<i>Nd-n</i>		91	<p>The nodes were created for key junctions with the parameters mentioned in <i>Chapter 3, section 3.4</i>.</p> <p>The Nodes were not assigned dedicated Controllers in the simulation as they were controlled by Control Agents hosted on the associated Grid Controller.</p>
Link	<i>Lnk-n</i>		67	<p>The links were created for representing the roads for linking the above Nodes, and used for the formation of routes. Links were also not assigned dedicated Controllers in the simulation as they were controlled by relevant Control Agents hosted on their Grid Controller. All the information such as local data and parameters were stored on their host Grid Controller.</p> <p>(The link on the simulation application appears Purple as Normal, Orange as congested and Red as Blocked)</p>
Traffic Light	<i>CtrlTL-n</i>		182	<p>Traffic Light/Signal Controls were simulated using the MADF based Control Application (described in <i>Chapter 7, section 7.3.2.4</i>) on a combination of PDAs and emulators. The 'Traffic Light Control' applications were associated with the relevant 'Traffic Light Controllers' application at communication and database level.</p> <p>The locations of the Traffic Lights were configured similarly to the current real locations.</p>
Traffic Lights Controller	<i>CtrlTL-n</i>		36	<p>The Controllers for the traffic lights controls were simulated using the MADF based Controller Application (as described in <i>Chapter 7, section 7.3.2.4</i>) on a combination of PDAs' and emulators' Controller Applications.</p> <p>Each Traffic Light Controller was configured to control about 5-8 traffic Controls based on their distances and routes.</p> <p>Each Traffic Light Controller hosted a Control Agent for its operations in the Agent based networked environment (Control Agents mentioned in <i>Table 8.2</i>).</p>

Vehicle Count Sensor	<i>cVC-n</i>		173	<p>These detectors sensors were placed on all crucial in- and outbound routes of the Links representing the current inductive loops in Coventry City.</p> <p>The sensors were simulated using a simulation utility, part of the Grid Controller Application (mentioned in <i>Chapter 7, section 7.3.2</i>)</p>
Vehicle Count Sensors Controller	<i>CtrlVC-n</i>		44	<p>Several Vehicle Count Sensor Controllers were configured for controlling multiple 'Vehicle Count Sensors'. The Controllers' applications were configured on PDA emulator software connected with the Grid Controller system.</p> <p>The Sensor Controller applications hosted a relevant Control Agent for its operations and communication with other Control Agents in the platform.</p>
Variable Message Display Board	<i>cMsgD-n</i>		23	<p>Variable Message Display Boards are not currently in use in the Coventry City area, but simulated versions were introduced on key diversion points in the system.</p> <p>These simulated boards were developed to evaluate the capabilities of the control type and to assess their use within the research.</p> <p>The boards were simulated using the MADF based Control Application (described in <i>Chapter 7, section 7.3.2.4</i>) on a combination of PDAs and emulators. The Variable Message Display Board Control applications were associated with the relevant 'Variable Message Display Board Controllers application.</p>
Variable Message Display Boards Controller	<i>CtrlMsgD-n</i>		6	<p>A Controller application was configured to Control a set of Variable Message Display Boards Controls in a Grid.</p>
Dynamic Traffic Sign	<i>cSgnD-n</i>		49	<p>Dynamic Traffic Signs are not currently in use in the Coventry City area, but are supported and recommended in ITS@CU. These signs are very flexible and provide various traffic signs as part of the ITS solution such as speed, directions and parking signs.</p> <p>Part-time one-way signs were simulated for limited routes where possible and no parking signs were</p>









				<p>simulated on other routes to allow more traffic.</p> <p>The locations of the Dynamic Sign Controls were chosen based on their relevance in terms of practicality (whether the sign was in a position where drivers could make use of the information) and importance (whether the road network benefited significantly from the placing of the sign).</p>
Dynamic Traffic Signs Controller	<i>CtrlSgnD-n</i>		16	Similarly a Controller is configured to control the associated Dynamic Sign Controls (described above) using the MADF based Control Application.
Other Controllers				Other supported controllers used in some of the test studies
Vehicles	<i>CtrlVeh-n</i>	 		<p>Different types of supported vehicles (normal, emergency and transport) were used in the test studies. Some Vehicle Controls were simulated using MADF based Vehicle Controller Applications (described in <i>Chapter 7, Section 7.3.3</i>) however most of the vehicle location data (GPS) and road occupancy data (Count Sensor data) was generated using the simulation utilities (discussed in <i>chapter 7, section 7.3.1.3</i>) which allowed real-time data manipulation to test different traffic scenarios.</p> <p>The number of vehicles and traffic flow values were configured using simulation data and they differed for different studies.</p>
Bing Map External Services	<i>SvcBing-1</i>		1	Microsoft Bing MAP service was used for Reverse geocoding and direction features (described in <i>Chapter 4, Section 4.1.3</i>). This Service was Controlled by the Service Agent (Service Agents mentioned in <i>table 8.3</i>)
Weather Information External Service	<i>SvcWS-1</i>		1	A test web service application was developed for simulations which provides similar functionality but with pre-configured data which can be altered in real-time to simulate different weather conditions. The web service was hosted on the Gateway web server.
Route Information External Service	<i>SvcRt-1</i>		1	A simulation purpose-built Web Service application providing route information data which can be configured to the street level.

Table 8.1: Entities (traffic Elements, Controllers and Controls) used in test studies/simulation

The following number of Control Agent types/subtypes (discussed in *chapter 4, table 4.2*) were configured for test/simulation studies (in Agent-logic mode):

 Control Agents		
Types/Roles	#	Description
<i>cAgtCtrGrid-n</i> for Grid Controller	6	Each Agent was hosted on the relevant Grid Controller (<i>CtrGrid-n</i> mentioned in <i>table 8.1</i>) providing the Agent-Logic for grid operations (described in the <i>chapter 4, table 4.2</i>). 6 Agents in total were setup for the study area during the simulation.
<i>cAgtCtrTL-n</i> for Traffic lights Controller	15	Provides Agent-logic to the host Controller to dynamically adapt the signalisation of the associated controls based on the traffic situation. A single Traffic lights Agent is hosted on 2-3 Traffic Lights Controllers (<i>CtrTL-n</i>) depending on distance.
<i>cAgtCtrSgnD-n</i> for Dynamic Signs Controller	8	Provides Agent-logic to the host Controller to dynamically display traffic signs (variable Speed, parking and direction signs) on the associated controls based on the traffic situation. This Agent is hosted on 1-2 Controllers (<i>CtrSgnD-n</i>) depending on distance.
<i>cAgtCtrVC-n</i> for Vehicle Count Sensors Controller	13	Hosted on 2-3 controllers (<i>CtrVC-n</i>) based on link length to provide Agent-Logic to the hosts
<i>cAgtCtrMsgD-n</i> for Variable Message displays Controller	6	1 Agent per grid to control all Variable Message Display Board Controls with a grid. (hosted on all <i>CtrMsgD-n</i> within a grid)
<i>cAgtCtrLnk-n</i> For Links	21	Multiple links were controlled by the link Control Agent which was used in limited studies. Similar to node agents, the number of links controlled by a single Agent varied in different test cases. The Agents were hosted on Grid Controller applications which allow simulating Link Controllers. The link Control Agents constantly monitors its link occupancy values and traffic flow rate (using the info provided by <i>cAgtCtrVC-n</i> Agents)
<i>cAgtCtrNd-n</i> For Nodes	29	Multiple nodes were controlled by this Agent and used in limited test studies. The number of nodes controlled by a single node Control Agent varied in different test cases. The Agents were hosted on Grid Controller applications which allow creating simulated mode Node Controllers.

<i>cAgtCtrCc-1</i>	1	Main Agent controlling the <i>CtrCc-1</i> (mentioned in <i>table 8.1</i>)
--------------------	---	--

Table 8.2: Control Agents used in test studies/simulation

The following number of Service Agent types/sub-types (discussed in *chapter 4, table 4.5*) were configured for simulation studies:



 Service Agents		
Service/Role	#	Description
<i>sAgtSvcBing-1</i> for Bing Map Service	1	Accesses Bing Service in response to other Agents' requests
<i>sAgtSvcWs-1</i> for Weather Service	1	Accesses Weather Information Service and interacts with Agents within the platform Hosted on a central control Application
<i>sAgtSvcRt-1</i> for Route Service	1	Accesses Route Information Service and interacts with Agents within the platform Hosted on a central control Application

Table 8.3: Service Agents used in test studies/simulation

All these external services were hosted on the Central Control System Web server (discussed in *chapter 7, section 7.3.1.2*) and the Service Agents were hosted by the “platform intelligence layer application” on the Application Server (discussed in *chapter 7, section 7.3.1.3*).

The following number of Operational Agent types/sub-types (discussed in *chapter 4, table 4.7*) were configured for simulation studies:

 Operational Agents		
Types/Roles	#	Description
<i>oAgtArb-n</i> Arbitrator	11	Several Arbitrator Operational Agents were used at different levels. <ul style="list-style-type: none"> • 2 for <i>CtrCc-1</i> (Central Control System) • 3 for <i>CtrGrid-1</i> (Grid-1 Controller)

		<ul style="list-style-type: none"> • 2 for <i>CtrGrid-3</i> (Grid-3 Controller) • 1 each for the other four Grids (<i>CtrGrid-2</i>, <i>CtrGrid-4</i>, <i>CtrGrid-5</i> and <i>CtrGrid-6</i>)
<i>oAgtSec-n</i> Security Agent	7	<p>Various Security agents were set up for authenticating/verifying Agents and their level of access at different levels:</p> <ul style="list-style-type: none"> • 1 at <i>CtrCc-1</i> (Central Control System) • 1 at each Grid Controller
<i>oAgtCtrMgr-1</i> Host Controllers Manager	1	A single Host Controllers Manager Agent was set up for managing Controllers within the platform during evaluation. Hosted on both Central Control and Grid level hosts.
<i>oAgtcAgtMgr-1</i> Control Agents Manager	1	An Agent was set up to manage all the Control Agents within the platform. Hosted on Grid level hosts.
<i>oAgtSvAgtMgr-1</i> Service Agents Manager	1	An Agent was set up to manage all the Service Agents within the platform. Hosted on Central Control.
<i>oAgtOAgMgr-1</i> Operational Agents Manager	1	An Agent was set up to manage all the Operational Agents within the platform. Hosted on both Central Control and Grid level hosts.
<i>oAgtSvcReg-1</i> Service Registry	1	A single agent was configured on the Central Control (Services bus/layer) to assist in service discovery and dynamic services composition in the platform.
<i>oAgtQoS-1</i> Agent QoS	1	This agent was set up on the Central Control System to check the Quality of Service (QoS) of all the Agents in the platform using the QoS values (described in <i>chapter 4, table 4.7</i>)
<i>oAgtPoolMgr-1</i> Pool Manager Agent	1	<p>This Agent was used in limited studies to evaluate the load balancing/sharing features of Operational Agents. It was only used for the two Arbitrator Agents in <i>CtrCc-1</i>.</p> <p>The agent ID (<i>oAgtPoolMgr-1</i>) of the Pool Manager agent is just an alias of the actual agent (<i>oAgtArb-1</i>) serving as a current pool manager in the cluster of the agents (<i>oAgtArb-1</i> & <i>oAgtArb-2</i>) in that pool.</p>
<i>oAgtSys-n</i> System Agent	3	Several System Agents were configured to perform log clean up and orphan messages clear-up tasks in the platform for evaluation purpose

Table 8.4: Operational Agents used in test studies/simulation

Remark: A different number of Agents and entities were used in different test studies (simulation configurations/setup).

The platform Agents mentioned in above tables for the simulation studies were organised according to the Agent Organisation structure described in *Chapter 4, Section 4.2*. They were split into “System” and “Grid” agencies.

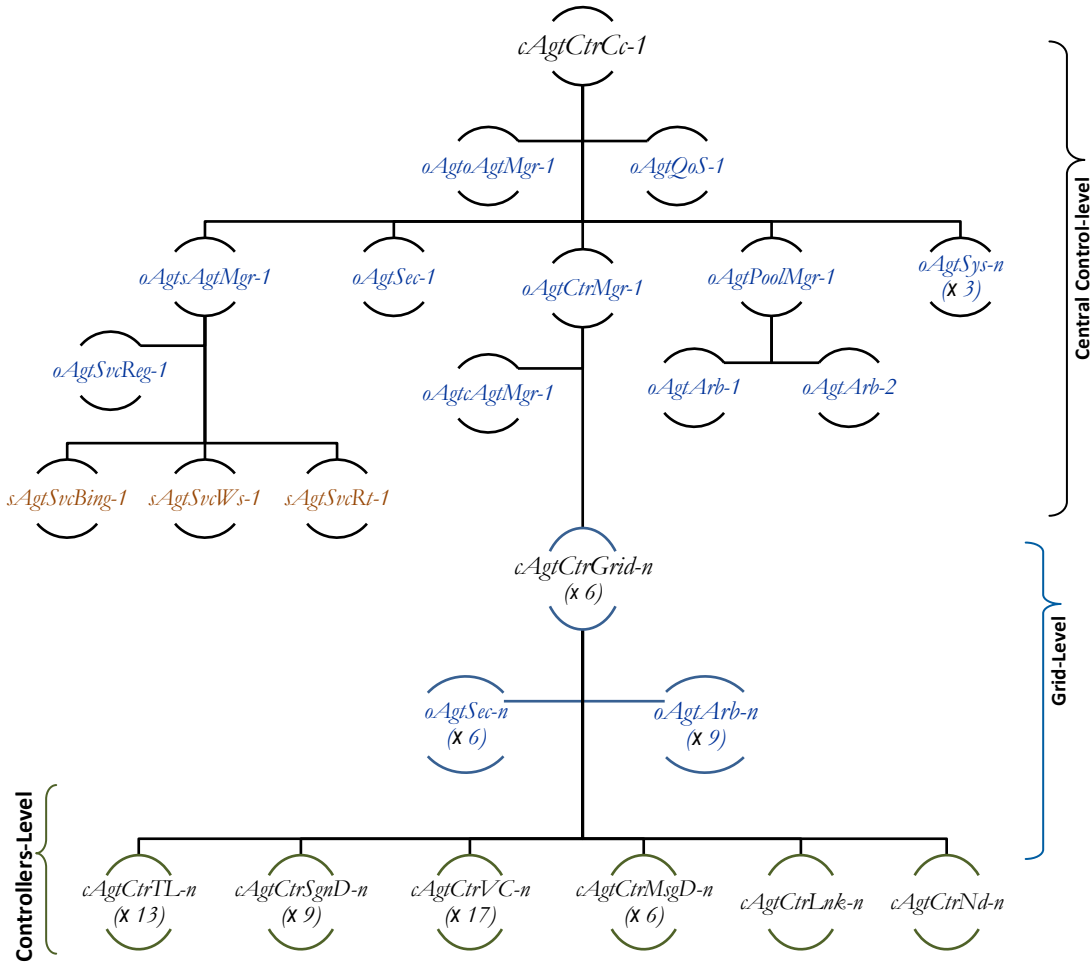


Figure 8.2: Organisational Structure of the platform Agents (for simulation)
(The Agent types and levels are shown in different colours)

The above diagram shows the organisational structure (hierarchy/levels) of these Agents, which is important in understanding the results and conclusions of the test cases.

8.1.4. Test data

The test data used for the evaluation primarily consisted of the historical traffic data (provided by UTM, Coventry City Council generated from their SCOOT system). The data was then adapted in order to perform wide ranging tests simulating different traffic situations. The data obtained from the SCOOT system included traffic flow (vehicles/hour) for different junctions and routes at different times and days. It also included the congestion rate, number of stops, average delays, roads saturation/occupancy percentages, inductive loop detectors flow and occupancy and stage lengths. Each study/test case describes the specific data used.

See “*Appendix L*” for a SCOOT report outlining the collected traffic data and “*Appendix J, CD*” for the bulk test data files.

8.2. Study cases and result analysis

This section presents an analysis of the different aspects of the research approach and their benefits. It presents the selected traffic scenario with different test cases (configurations/setup) in order to show how they achieve the research objectives especially the communication, coordination and decision making behaviour of Semantic Agent based Controls in SOA based distributed setup.

All the studies and cases are based on the platform elements, Agents, test data and study area described in *section 8.1*.

8.2.1. Problem detection & response (Incident scenario)

In this scenario, a road block representing an incident completely stopping the traffic flow on a busy road (near Junction 10314 – downstream) was simulated. The blockage points (road blocks) in the real world are variable (i.e. dependent on the situation with different severity) therefore in this study a complete blockage (no passing traffic) was simulated on the affected link in a certain direction.

The road block affects 6 routes/streams of traffic flow causing congestion on major routes in *Grid-3* going towards *Grid-1* (the city centre grid), see *figure 8.3*.

This particular junction was selected due to its importance and also as it does not have any major immediate alternative routes apart from a residential street. This part of the junction historically causes bottlenecks especially during the morning rush-hour for the traffic flow towards the City centre. Therefore, the data used for this evaluation was morning rush hour traffic data between 08:15 and 08:30 AM on a Monday (see “*Appendix L*”).

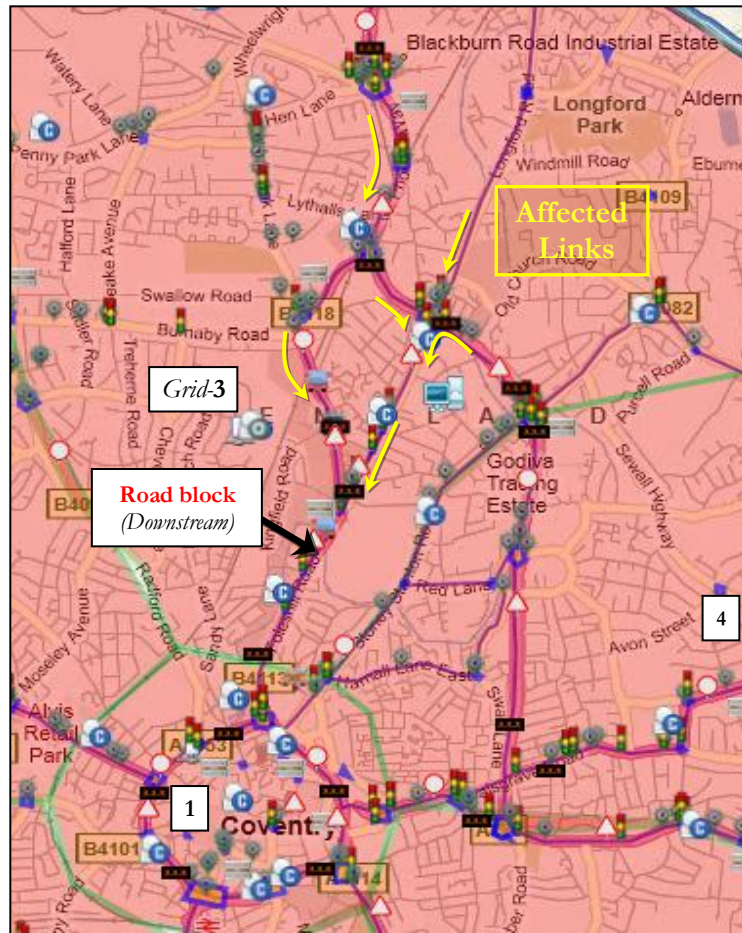


Figure 8.3: Affected links in *Grid-3* due to the road block/incident
(Image section from the simulation view shown in *figure 8.1*)

With the introduction of such an incident/road block, the Agents in ITS@CU were configured to detect the problem by coordinating with each other using the Ontologies (described in *chapter 5*) and act appropriately using the Rules and Plans (described in *chapter 6*). *Figure 8.4* outlines the behaviour of the ITS@CU platform and the steps that the Agents should perform in this situation as per the platform's current configuration (defined by the Rules, parameters and Ontologies). All these ontologies and rules used in this study are provided in "*Appendix J, CD*".

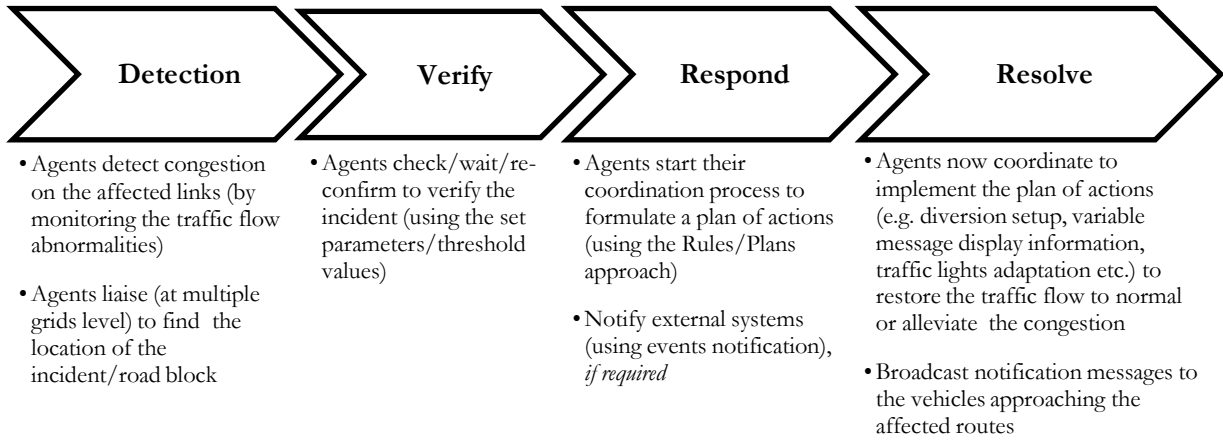


Figure 8.4: ITS@CU behaviour overview for traffic problems detection and response

(Showing the steps in the context of the situation described in the scenario overview)

Methodology

This scenario/study was simulated in different configuration setups each with a different number of elements, controls and Agents (detailed in *table 8.1, 8.2, 8.3 and 8.4*). Each simulation setup was performed 10 ten times/runs (with different sets of parameters and threshold values) in both “Agent-Logic” and Default-Logic”.

The original test data was also altered in order to assess different aspects of the system especially the behaviour of Semantic Agent based Controls in SOA environment.

Remark: Using the Controllers in Default-Logic mode (configurable in the Controller simulation application, mentioned in *chapter 7*) disables the Control Agent governing the Controllers’ operations, hence it completely simulates the behaviour of a normal fixed traffic Controller. The Controller in default mode still uses the same communication layer and communicates with Operational Agents but without the semantic-content and the intelligence provided by Control Agents. It is also not capable of communicating with other Controllers hence the communication becomes more centralised and the complete communication at the Controller-Level of the organisational structure (described in *figure 8.2*) becomes non-functional.

Results

The following presents an analysis of how the platform Agents (Control Agents in particular) actually behaved/performed when the road block was introduced during the simulations studies.

Remark: Configuration setup 1 is presented in detail below. Since the other configurations were similar to setup 1, only the differences, the result data and the analysis are presented in this section.

In this scenario, the following Agents were primarily involved in the detection and response process. Please note that various other Agents (mentioned in *table 8.2*, *8.3* and *8.4*) were also involved, performing their routine functions, however the following Agents were directly involved or were part of the decision making process in different stages mentioned in *figure 8.4*.

Agents	Details
<i>cAgtCtrGrid-3</i>	for <i>CtrGrid-3</i>
<i>cAgtCtrGrid-1</i>	for <i>CtrGrid-1</i>
<i>cAgtCtrGrid-4</i>	for <i>CtrGrid-4</i>
<i>oAgtArb-1</i>	Main Arbitrator Agent for <i>CtrCc-1</i> providing arbitration between Grid level Agents and communication initiation. It is also set as a pool manager for load balancing.
<i>oAgtArb-3</i>	Arbitrator Agent for <i>CtrGrid-1</i>
<i>oAgtArb-6</i> , <i>oAgtArb-7</i>	Two Arbitrator Agents in <i>CtrGrid-3</i>
<i>oAgtArb-8</i>	<i>CtrGrid-4</i> Arbitrator Agent
<i>sAgtSvcBing-1</i>	Service Agent responsible for external route/geocoding service (MS Bing Service)
<i>sAgtSvcWS-1</i>	Service Agent responsible for Weather Info external service
<i>cAgtCtrVC-n</i> (x 4)	4 Control Agents (with 12 <i>CtrVC</i> and 49 <i>cVC</i>) in <i>CtrGrid-3</i>
<i>cAgtCtrTL-n</i> (x 4)	4 Control Agents (9 <i>CtrTL</i> and 43 <i>cTL</i>) in <i>CtrGrid-3</i>
<i>cAgtCtrMsgD-3</i>	1 Control Agent for the <i>CtrMsgD-3</i> (with 6 <i>cMsgD</i>) in <i>CtrGrid-3</i>
<i>cAgtCtrSgnD-n</i> (x 2)	2 Control Agents (4 <i>CtrSgnD</i> and 15 <i>cSgnD</i>) in <i>CtrGrid-3</i>
<i>cAgtCtrLnk-n</i> (x 3)	3 Control Agents (23 links) in <i>CtrGrid-3</i>
<i>cAgtCtrNd-n</i> (x 5)	5 Control Agents (37 nodes) in <i>CtrGrid-3</i>

Table 8.5: Agents involved in detection and response in the scenario (*Configuration Setup 1*)

In the “**Detection**” step, the relevant Vehicle Count Controller Agents $cAgtCtrlVC-n$ observed and reported a low traffic flow rate from vehicle count sensors on the affected links (the alarm for abnormal values was set to occur if the flow rate dropped below 100 vehicles per hour for all $CtrlVC$). The Link Controller Agents $cAgtCtrlLnk-n$ involved calculated a high occupancy (the alarm for abnormal values was set to occur if the occupancy average was above 70% for a link).

Next, the $cAgtCtrlLnk-n$ Agents checked with the adjacent/corresponding Links and Nodes by broadcasting request messages to check their occupancy status. During this process the specific link with abnormal occupancy was identified (using the algorithm in links’ Agent-logic to check Links where the traffic flow from an incoming node is considerably different from the outgoing node). Then the $cAgtCtrlVC-n$ agents associated with the identified link assessed their $CtrlVC-n$ flow data to identify the location of the blockage (using a similar algorithm in $cAgtCtrlVC-n$ agent-logic which checks the flow rates from the $cVC-n$ detectors to find a cVC showing abnormal values or a different flow rate compared to adjacent sensors/detectors).

At this point, the $cAgtCtrlLnk-n$ Agents involved reported the incident location to the local grid level Arbitrator Agent ($oAgtArb-6$) triggering the “**Verification**” step. The Arbitrator Agent monitored the situation for a period of time (the waiting time was set to 5 minutes) by repeating the status check and verification process every 30 seconds (with relevant $cAgtCtrlLnk-n$ and $cAgtCtrlVC-n$) to confirm if the situation remained the same or degraded further in order to verify and establish the incident before escalating it.

Once the incident was verified the $oAgtArb-6$ triggered the “**Respond**” step by sending a request to Arbitrator Agent at Central Control Level ($oAgtArb-1$) to propose a Plan. $oAgtArb-1$ liaised with other Grid Arbitrator Agents (in this case $oAgtArb-3$ and $oAgtArb-8$) to check their status in order to formulate a Plan. In this scenario only *Grid-3* was affected by the initial congestion but (due to the Rules configuration set in the simulation) *Grid-1* and *Grid-4* were also involved in the response process of managing a traffic diversion to alleviate the congestion. A diversion involves checking the status of all the Nodes within a set proximity (the range was set to 100 meters on the adjoining links on the affected Nodes) in order to identify the Nodes and links which can best support extra traffic flow.

Once the Nodes were identified the $oAgtArb-1$ Agent used the service agent $sAgtSvcBing-1$ to provide an alternative route generation for the affected roads. The $sAgtSvcBing-1$ agent not only provided the fastest route at node level (provided by the Bing Maps Service) but also liaises with the weather service Agent $sAgtSvcWS-1$ using the composite services flow feature of the platform’s service bus layer to find the most appropriate alternative route for the current circumstance. In the simulations, the $sAgtSvcWS-1$ agent was set to a value indicating no weather alarms.

In the “**Resolve**” step, the Arbitrator Agent on the Central Control Level *aAgtArb-1* requested that the involved Arbitrator Agents at the Grid Level to execute the relevant steps of the plan for implementing the diversion (as recommended by *sAgtSvcBing-1* – which was separate for different routes, as shown in figure 8.5 and 8.7). Next the individual Arbitrator Agents took the appropriate actions at their grid level (*aAgtArb-6* for Grid-3, and *aAgtArb-8* for Grid-4).

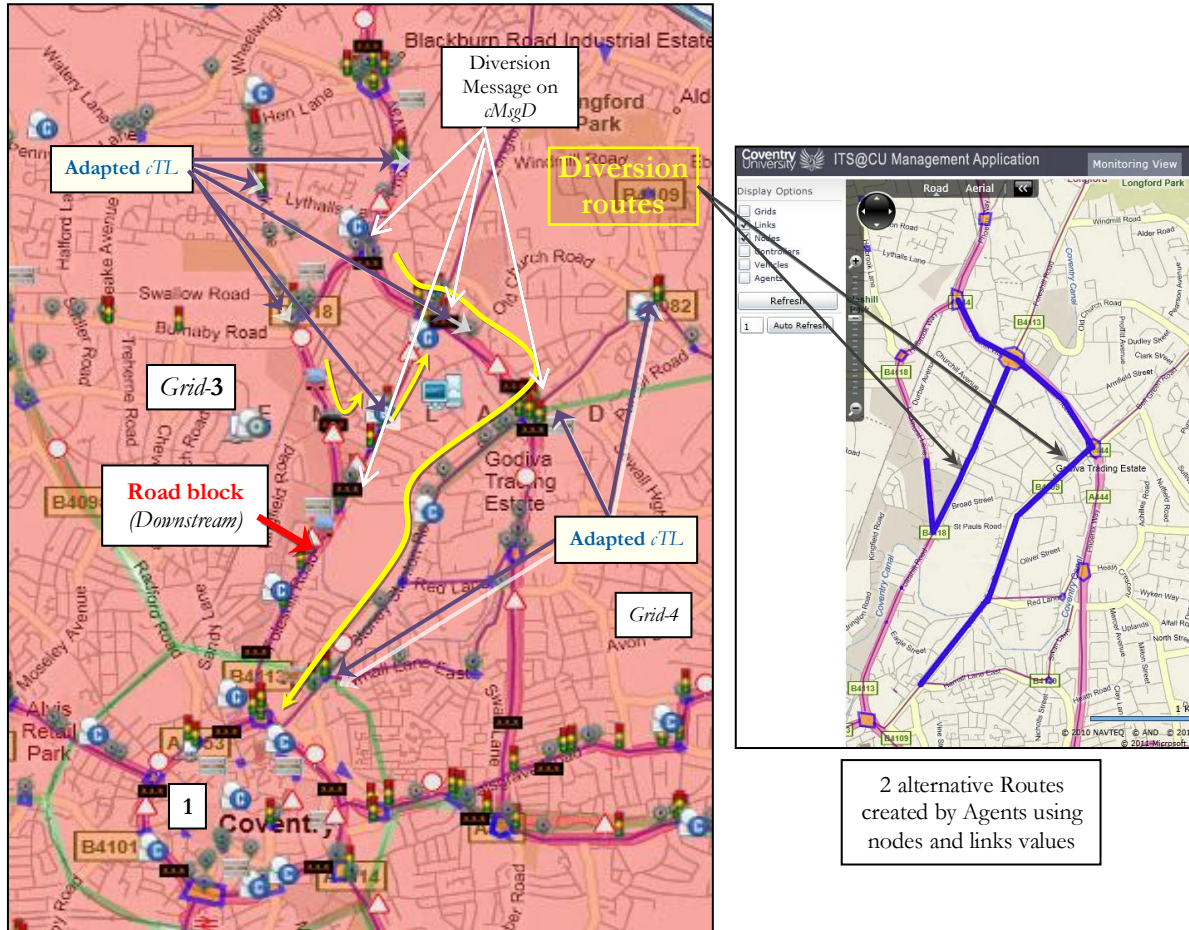


Figure 8.5: Route diversion, Traffic lights adapted and Variable Messages

As per the configuration (in the Ontologies and Rules used), *aAgtArb-6* and *aAgtArb-8* sent request commands with the specific set of Rules to the relevant Traffic Lights Control Agents (*cAgtCtrTL-n*) and Variable Message Display Control Agents (*cAgtCtrMsgD-n*), which were on the affected or diversion route links (see figure 8.5). Each *cAgtCtrMsgD-n* agent interpreted the request message instructions and executed the required actions described in the semantic-content of the message on their host controllers i.e. to display the road block and diversion messages for a set period of time (10 minutes in the simulation). Similarly, each *cAgtCtrTL-n* agent executed the message instructions on their host controllers i.e. adapting the associated traffic lights controls by adjusting green light time on the affected/diverted links (the value was set to $\pm 8\%$ recommended by UTMCO, Coventry Council and used in the SCOOT system) thereby allowing extra vehicles on diverted routes.

The modified values and status (on those $cAgtCtrl-n$ and $cAgtCtrlMsgD-n$) were set for a limited time period (10 minutes in this configuration) after which the values returned to their default values/status. This timeframe is configurable and the $aAgtArb-1$ Agents can repeat the process of requesting $aAgtArb-6$ and $aAgtArb-8$ to re-apply the same Rules, however in this simulation this option was not used.

Remark: At any given point if the situation changes i.e. occupancy values changes which means congestion no longer exist, the $cAgtCtrlLnk-n$ Agent notifies the $aAgtArb-6$ which notifies the $aAgtArb-1$. This way the request for Plan can be cancelled. The arbitrator Agents use the conversation-ID and the Agent-list in the message instruction to send cancellation message to the involved agents (mentioned in *chapter 5*).

Additionally, in this scenario the Grid Controller Agent ($cAgtCtrlGrid-3$) was also configured to send broadcast message (congestion ahead and the blocked junction) to the vehicles (PDAs) approaching the congestion (using their current GPS location/coordinates in proximity of 70 meters of the affected links based on the flow direction as mentioned in *chapter 7*).

The simulation results at different observation points (mentioned in *result analysis* section below) were recorded. Each simulation run ended with the Resolution Time (RT) observation (which indicates that the traffic flow has been restored to within the normal range of values).

The following flowchart diagram further explains the steps and the behaviour of the involved Agents in this scenario result description.

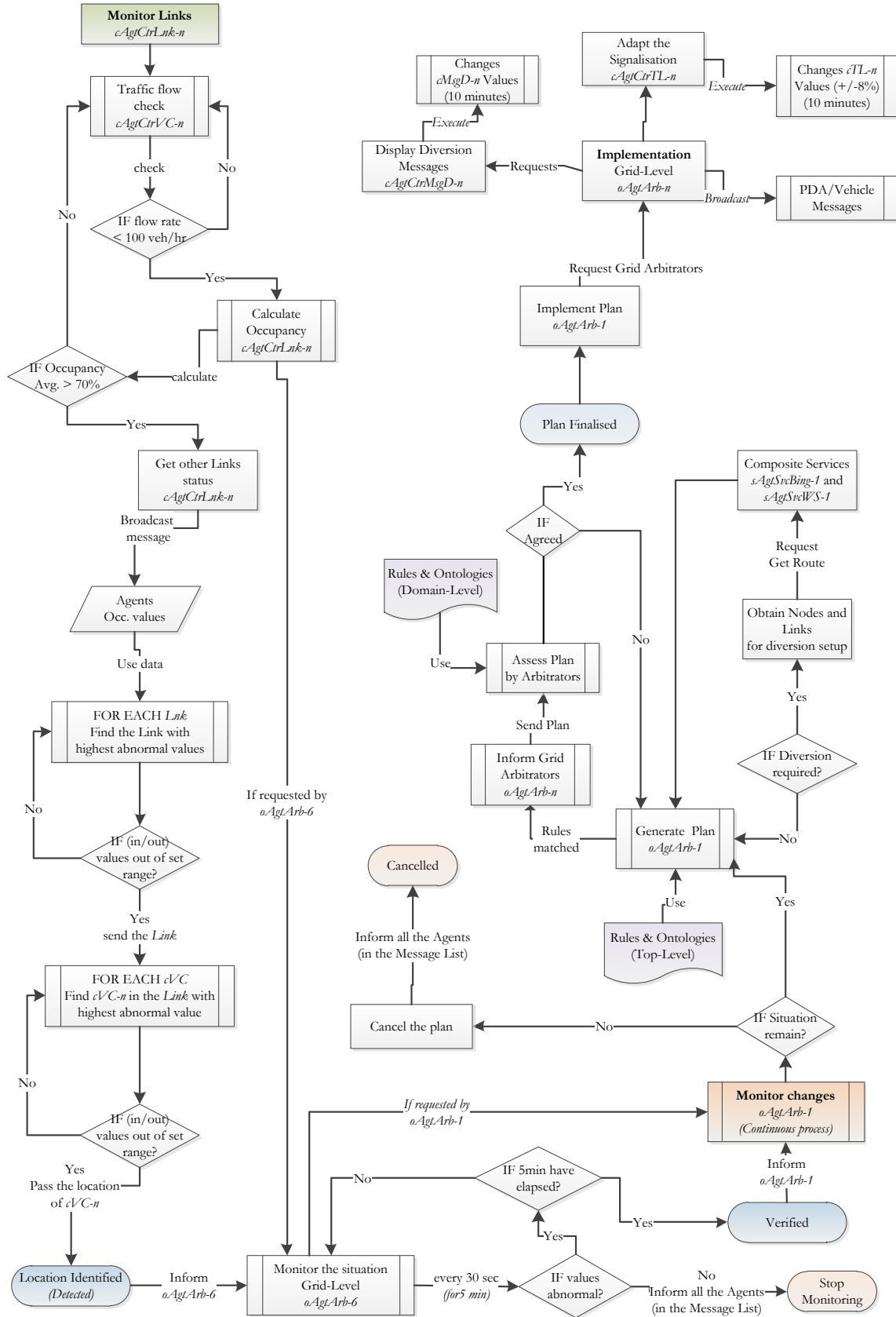
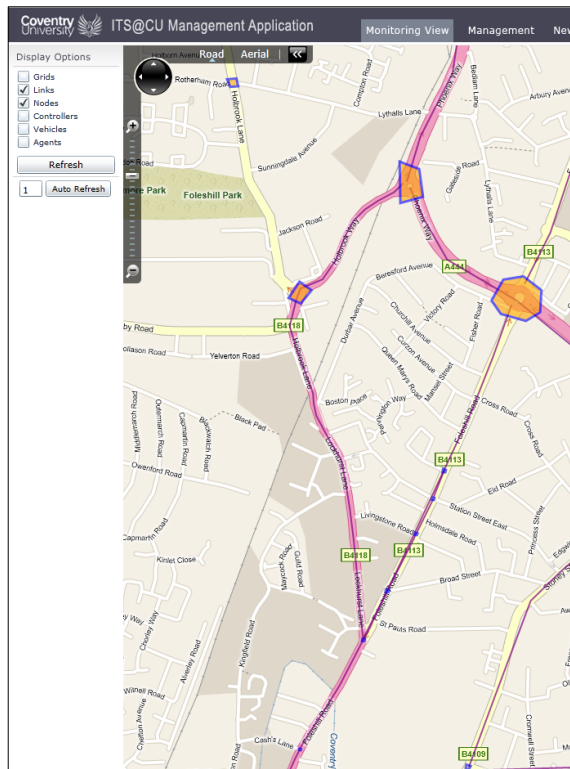


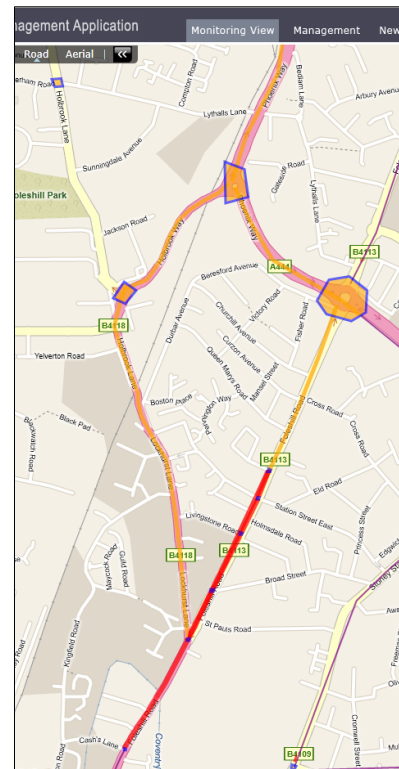
Figure 8.6: Result: Agents behaviour representation in flow chart

All the Ontologies and Rules used, and the Plan generated in this study are provided in “Appendix J, CD”.

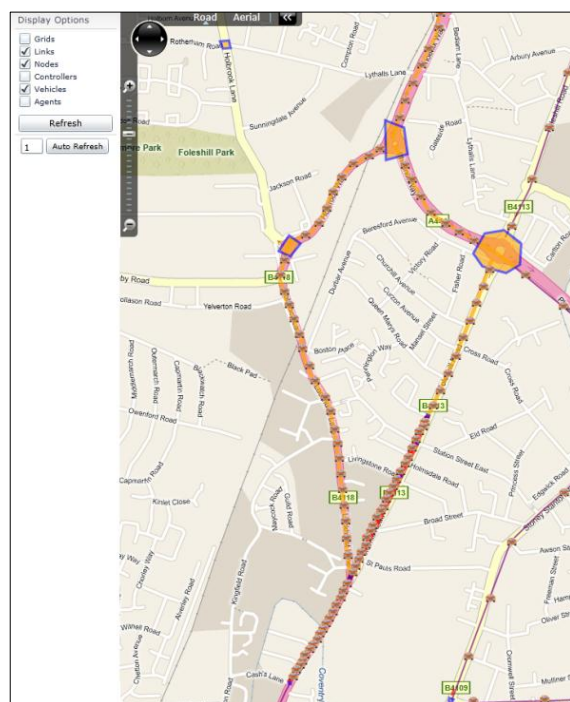
The following simulation views (application screens) show the underlying behaviour/outcome of the Agents involved in this scenario.



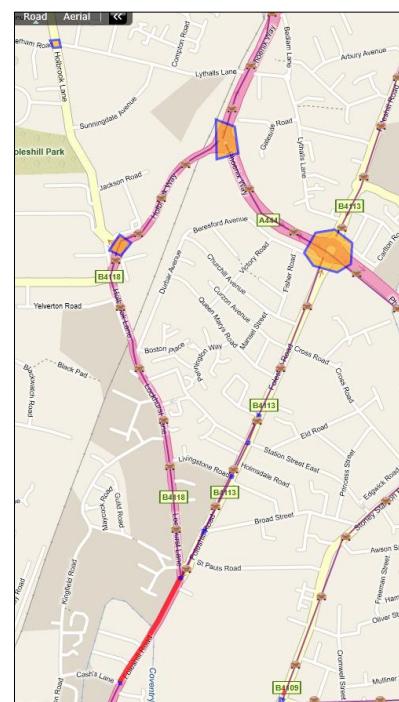
Simulation view 1: Before the incident was introduced.
Links are normal in Purple colour lines



Simulation view 2: After the incident was introduced.
(Red Links with abnormal flow rate and Orange links with flow rate value below normal threshold)



Simulation view 3: Same as view 2, but showing the simulated vehicles on the affected link. (Each link has different vehicle density as per its current flow rate)



Simulation view 4: After the resolution Plan implementation. The main blocked link remained blocked even after the Resolution Time (RT).

Figure 8.7: Simulation view of the scenario and its result

Result analysis

The results of this study scenario were analysed to assess the following key areas:

Congestion/Incident detection: The detection behaviour of the platform’s approach was assessed in terms of:

- Detection Report Rate (DRR): The rate of problem detection alarms by Control Agents
- False Alarm Rate (FAR): The number of incident free intervals with false incidents alarms divided by the total number of incident free intervals
- Mean Time To Detect (MTTD): The difference between the time of accident occurrence and the time of accident detected/verified
- Detection Rate (DR): The actual incidents verified (DRR - FAR)

The following table outlines the setups/configurations used in this traffic scenario:

Setup 1	The main scenario (configuration as described above)
Setup 2	The same as Setup 1, but with 3 road blocks on separate links
Setup 3	The same as Setup 1 but with three times higher concentration of Control Agents, Controllers and Controls (as per the same Agent/Controller ratio in <i>table 8.2</i>)
Setup 4	A combination of Setup 3 (Three times higher concentration) and Setup 2 (with 3 road-blocks)

Table 8.6: Configuration setups (1-4) used in the “detection and response” scenario

Each simulation run in the above configuration setups used the test data from the morning rush hour traffic data between 08:15 and 08:30 AM on a Monday (see “*Appendix L*”). The vehicle flow on average for that junction at that time was 1209 vehicles/hour.

The following table presents the results obtained and calculated from all the setup runs:

Configuration	# Links $c_{AggCtrlLink-n}$	# Nodes $c_{AggCtrlNd-n}$	# Detectors $AggCtrlVC-n$	Blockage Points	DRR % (Agent-Logic mode)	DRR % (Default-Logic mode)	FAR % (Agent-Logic mode)	FAR % (Default-Logic mode)	MTTD _{sec} (Agent-Logic mode)	MTTD _{sec} (Default-Logic mode)	DR % (Agent-Logic mode)	DR % (Default-Logic mode)
Setup 1	21	29	15	1	100	100	0	10	6	11	100	90
Setup 2	21	29	15	3	100	100	10	30	9	13	90	70
Setup 3	67	91	44	1	100	100	0	20	16	21	100	80
Setup 4	67	91	44	3	100	100	20	50	29	33	80	50

Table 8.7: Result data obtained relevant for detection evaluation

As each setup had 10 runs per mode, the percentage values are all multiples of 10.

Remark: The blockage points were selected to make sure the involved Control Agents picked up the traffic abnormalities as the main intention was to assess the Controllers behaviour in both Agent and default modes – not the strength of the incident detection algorithm, therefore the DRR% and DR% values in both modes in all setups were high (as seen in *table 8.7*).

The MTTD values in the result were the actual time duration of the detection processing (i.e. configured decision wait timers values were discarded by the stop watch module embedded in the simulation utility).

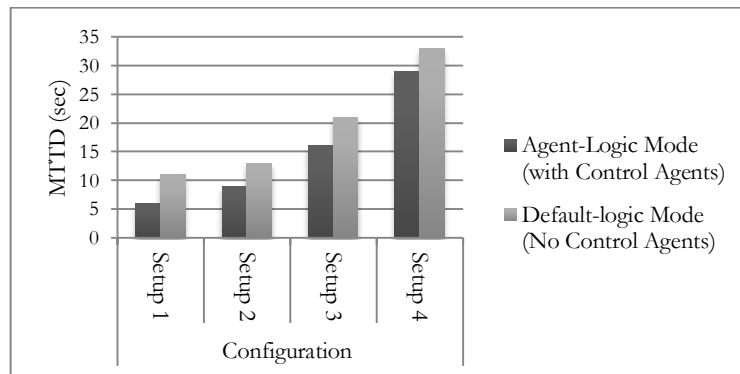


Figure 8.8: Mean Time To Detect (MTTD)

The Control Agents were quicker in detecting and verifying incidents in all setup runs as compared with default mode Controllers. When the number of Control Agents was increased (in setup 3 and 4), the MTTD values also increased due to the extra communication/transactions by the Control Agents.

One of the key findings based on the MDDT result values was that the ratio of the number Controllers per Control Agent is very important. The larger the number of Controllers per Control Agent the lower is the need for outside communication (Between the Control Agent with other Agents). However the drawback is that a single Control Agent has to control the operation of multiple Controllers and work in a split-site hosting mechanism (described in *Chapter 4*) which results in extra intra-agent level communication (between Controllers/hosts) and can have a negative impact on the host controller resources. The best approach is to balance this ratio depending on the capacity, resource, communication technology and distance between the Controllers.

In this study, the PDA emulators were used and the best ratio on average was 3 Controllers per Control Agent. However, this can vary depending on the type and capacity of the Controllers, so this has to be approached based on these factors.

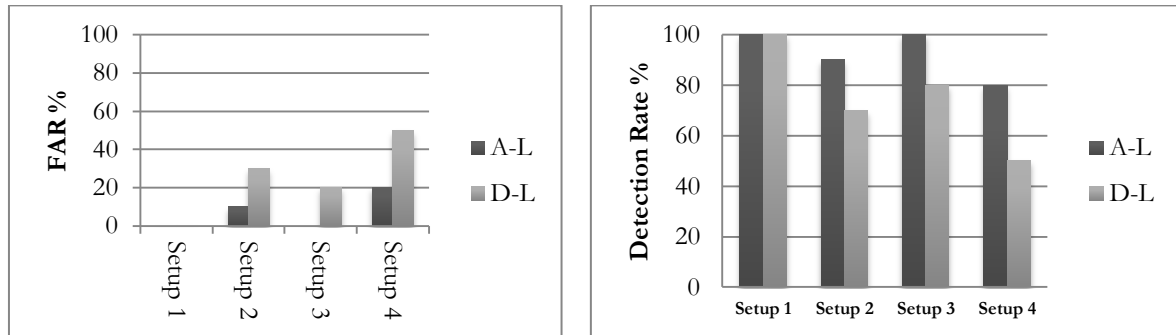


Figure 8.9: Left: False Alarm Rate (FAR) and Right: Detection Rate (DR)

A-L = Controller in Agent-Logic Mode

D-L = Controller in Default-Logic Mode (i.e. No Control Agent)

As seen in *figure 8.9*, FAR is where Agent based Controls have an advantage over fixed Controllers. Agent-based Controls raised much lower false alarms for the same runs when compared with Controllers in default mode. The main reason was that the Control Agents were configured to coordinate between each other and establish the incident location and the link where occupancy value was abnormal and only then report to the Arbitrator Agents (as described in the flow diagram, *figure 8.6*). This reduced the unnecessary detection reporting by Controllers to their Grid Arbitrator Agents, hence reducing the FAR.

Consequently, the DR (as seen in *figure 8.9*) was also much better due to low FAR in the Agent-based Control approach.

Congestion/Incident Response: The response behaviour of the platform's approach was assessed in terms of:

- **Mean Time To Respond (MTTR):** The difference between the time that the accident was verified and the time that the resolution Plan was generated and dispatched to Controllers for implementation (as mentioned in *figure 8.6*). The lower the MTTR, the better it is for the overall system performance.
- **Controllers Scope:** The number of Controllers (*CtrTL*, *CtrMsgD* and *CtrSgnD*) involved/affected by the resolution Plan i.e. how many controllers' property values were changed/adapted to return the traffic flow to normal. The wider the scope the better the adaptation that can be achieved.
- **Resolution Time (RT):** The time taken for the traffic flow on the Links (involved/affected in the resolution Plan) to become normal and stay normal for a period of time (10 min value was set in the simulation) without any further DRR (incident/congestion detection report). RT is used for estimating the overall effectiveness of the approach and the quicker the RT the better the approach.

In this study an additional setup/configuration was also included in the simulation runs in addition to the four configurations described previously:

Setup 5	Setup 3 and Setup 2 with the maximum Average traffic flow rate (2000 vehicles per hour in the involved links)
----------------	---

Table 8.8: Additional configuration (setup 5) used in the “detection and response” scenario

The following table presents the results obtained and calculated from all the setup runs:

Configuration	# Links $c_{AggCtrlLink-n}$	# Nodes $c_{AggCtrlNd-n}$	# TL $AggCtrlTL-n$	# MsgD $AggCtrlMsgD-n$	# SgnD $AggCtrlSgnD-n$	Blockage Points	Traffic Flow	MTTR $_{sec}$ (A-L)	MTTR $_{sec}$ (D-L)	Scope # CtrlTL (A-L)	Scope # CtrlTL (D-L)	Scope # CtrlMsgD (A-L)	Scope # CtrlMsgD (D-L)	Scope # CtrlSgnD (A-L)	Scope # CtrlSgnD (D-L)	RT mm:ss (A-L)	RT mm:ss (D-L)
Setup 1	21	29	15	6	8	1	1208	17	16	9	7	1	1	4	NA	3.4	6.5
Setup 2	21	29	15	6	8	3	1208	31	27	11	8	1	1	4	NA	6.2	8.3
Setup 3	67	91	44	18	14	1	1208	41	33	28	23	3	3	12	NA	8.3	13.2
Setup 4	67	91	44	18	14	3	1208	62	53	33	20	3	3	12	NA	11.3	23.2
Setup 5	67	91	44	18	14	3	2000	65	57	35	22	3	3	12	NA	15.4	27.4

Table 8.9: Result values relevant for response evaluation

Remark: The MTTR and RT values in the result *table 8.9* are the actual time of the response processing duration. Other duration values such as the pre-configured decision wait time values, connection setup and authentication/validation time were discarded in order to analyse the actual duration relevant for analysing the results.

As seen in *figure 8.10* (left), the MTTR values in both modes showed a similar trend i.e. the more controllers involved the higher the MTTR. Although the default mode was slightly quicker as it does not require Control Agents to be between the Grid Control (Arbitrator agents) and the Controller (as described in *figure 8.2*). The reason MTTR trends were similar in both modes is because the communication was mainly from grid arbitrator agent to controllers and that type of communication is not affected whether a controller is in default mode or in agent-logic mode.

In the Controllers Scope analysis (*figure 8.10*) the results show that agent logic is more dynamic. As the situation becomes more complex (for example an increase in the number of controllers involved or incidents/road blocks) the agent logic increases the scope of the involved controllers accordingly as compared to default mode which have a fixed scope.

The main reason for the wider scope in Agent-mode is due to the ability of Control Agents to communicate with each other, due to having sophisticated relationship ontologies with the other Control Agents at their Controller's level. If a resolution plan does not directly include a specific control agent,

but the associated agent can assist by involving other control agents then that agent is included in the coordination of the plan (e.g. if a resolution Plan includes a particular link and the control agent for the link believes that other links need to be involved to better implement the plan then it will start coordinating with the link agent). The better the relationship level ontologies (described in *chapter 4*) the more effective the scope management becomes.

The wider the scope of Controllers in implementing the resolution plan the more likely the resolution plan is to be effective however the wider the scope the longer it will take to form a consensus between the involved agents. So the domain-level Ontologies for describing Agents relationship must be designed using a balance approach to avoid any Scope related negative impact.

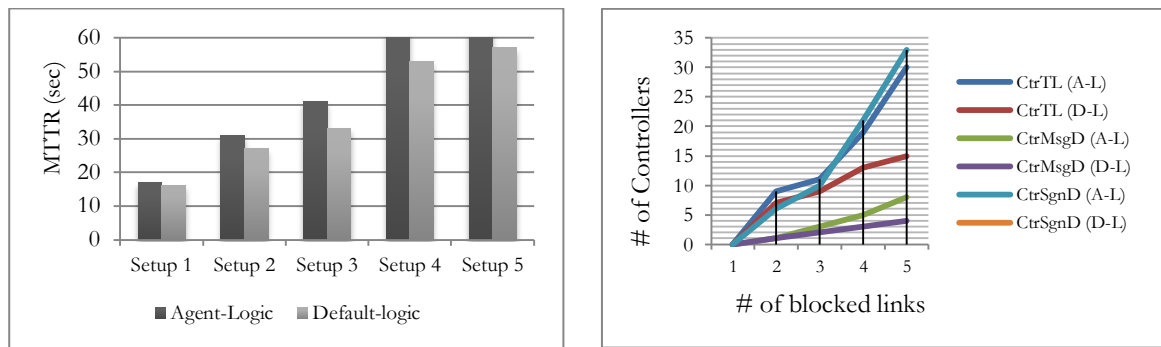


Figure 8.10: Left: Mean Time To Respond (MTTR) and Right: Controllers Scope

The main advantage of the Agent-based Controls was observed in the RT evaluation which shows the overall effectiveness of the approach. As seen in *figure 8.11*, the agent-logic mode helped the controllers quickly resolve the traffic flow due to the wider controller scope but mainly because Control Agents provide more functionality to their host traffic controllers (dynamic Traffic Lights adaptation, localised message displays to alerts vehicles etc.).

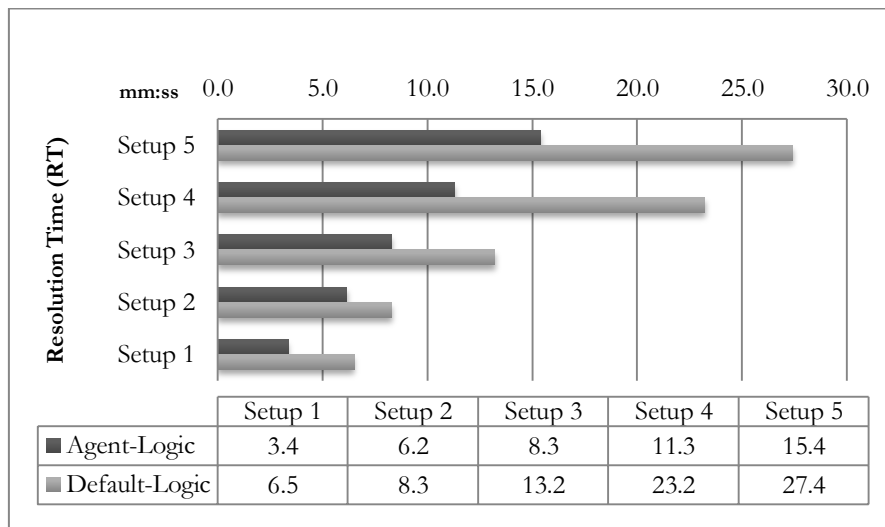


Figure 8.11: Resolution Time (RT)

The flow rates were also quickly normalised by agent-based controls as compared to fixed controllers (as demonstrated in *figure 8.12*). For example, in setup 1 and setup 3 simulation, the Agent-based Controls took around 20% less time to reach the RT (to restore the flow rate to 1800 vehicles/hour which was within the normal threshold value set for the simulation) on the affected links (Also seen in *figure 8.7*).

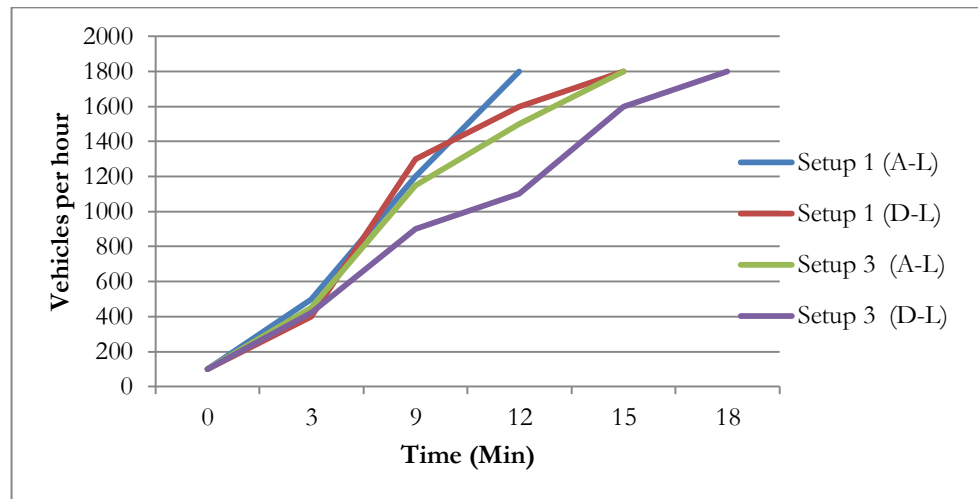


Figure 8.12: RT efficiency in normalising flow rate

Overall the quicker RT values indicate that Agent-Based Control allowed more vehicles to pass through the affected links over the period of time due to the Control Agents ability to implement the resolution Plan (on diversion routes) in a more dynamic way (*for example* better Controller Scope in Traffic Lights signal adaptation). Fixed Controllers can only react to the implementation plan in the hard coded manner with a fixed scope.

8.2.2. Communication approach evaluation

The communication approach of the Agent-based Controls in this research was focused mainly on reducing the number of transactions between Control Agents whilst optimising and enhancing the communication by embedding flexible semantic data within the agent's messages.

Transaction reduction: The Agent-based Controls approach reduces the overall number of transactions by significantly reducing the direct communication between traffic Controllers and Central/Grid control systems. The reduction of transactions is very important in a SOA environment (Daigneau, 2011; Erl et al., 2012) which involves geographically dispersed Services across networks. It is also important to curtail the processing overheads associated with XML messages (SOAP envelopes and embedded ontologies).

In a fixed control system (or default logic mode) each controller requires direct communication with a grid control or central control system to report its status or receive any commands. This is a massive communication overhead in a large ITS system due to the number of transactions involved, especially with the status messages between the controllers and central control system. In an Agent-based Controls approach, a single Control Agent governs multiple Controllers which reduces the level of communication between Controllers and Grid Agents/Systems (as one Agent communicates on behalf of all its Controllers) but most importantly the Control Agents coordinate with each other and perform decision-making at the Controller-Level to keep communication with Grid-Level Agents to a minimum (see the 'detection' steps of the incident detection scenario and in *figure 8.2*).

The chart in *figure 8.13* represents the number of transactions in the incident detection scenario (Setup 1), described in *section 8.2.1*. These transaction numbers were obtained from the simulation runs using the transaction logger module embedded within all the host central, grid and traffic controller/control applications (mentioned in *chapter 7, section 7.3.1.3, 7.3.2.1, and 7.3.2.4*).

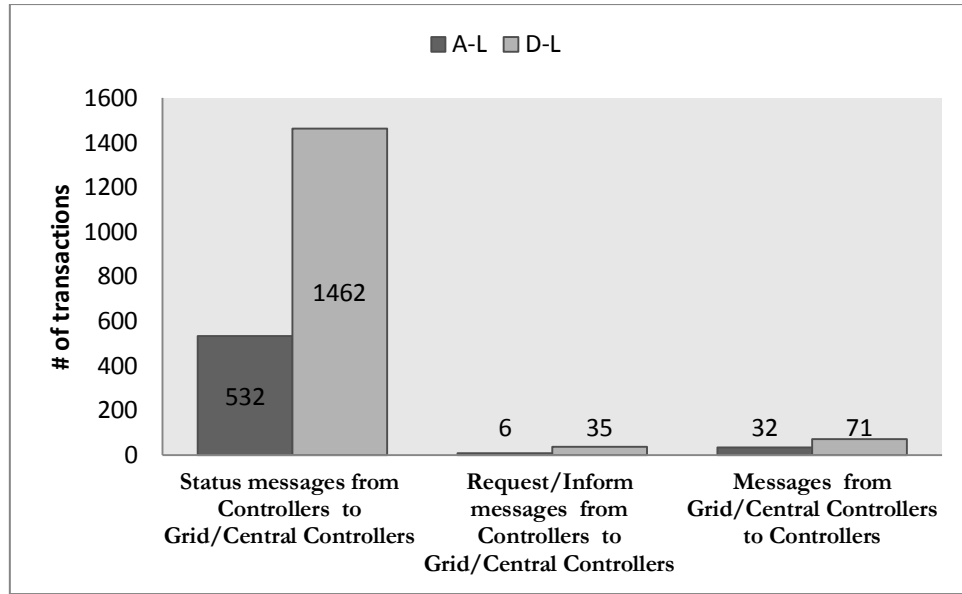


Figure 8.13: Transaction reduction between control agents at Controller-Level and Grid-Level

A-L = Controller in Agent-Logic Mode

D-L = Controller in Default-Logic Mode (*i.e.* No Control Agent)

Figure 8.13 shows a reduction in controllers' transactions between each level in Agent-based Control and default logic mode. In the incident detection scenario (Setup 1), there were 9 Traffic Light Controls (*Ctrl*) controlled by 4 Traffic Light Control Agents (*cAgtCtrl*) and the results show that the number of transactions for basic status reporting from Controllers to Grid/Central Control were reduced by more than half (532 as compare to 1462 transactions). This reduction was due to the Control Agent communicating with other Agents on behalf of all its Controllers rather than each Controller communicating directly with Grid/Central Control systems in a non-control agents based setup. Similarly, in the incident detection scenario (Setup 1), when the abnormality in the traffic flow rate was first detected by Vehicle Count Sensor Control Agents (*CtrlVC-n*), it started the communication between all the Control Agents involved (*cAgtCtrlVC-n* and *cAgtCtrlLnk-n*) to identify the incident location at the Controller-Level only. The incident was then reported to the Arbitrator Agent (*aAgtArb-6*) at the Grid-Level by a single Control Agent and only when the incident location was identified and confirmed. This further reduced the number of transactions between the Controller-Level and Grid-Level Agents (as shown in *figure 8.13*).

Remark: At the Controller-Level, Agent-based Controls require additional transactions between Control Agents (and its controllers). However, these transactions are between Control Agents which are closely interconnected at the Controller-Level as compared to the Central/Grid Control systems, which may be in different networks, geographic locations and may have bandwidth issues or host system limitation. So the reduction in transactions between Controller-Level agents and Grid-Level/Central Control-Level

agents (As demonstrated in *figure 8.2*) provides a key advantage to Agent-based control as compared to fixed traffic control systems.

Communication intelligence and flexibility: The semantic layer in Agent communication (mentioned in *chapters 5 and 6*) provides a high-level of functionality and flexibility in the form of Semantic-content, Ontologies and Rules embedded within Agent messages. In this way the agent messages can be either a simple request-reply between two Agents or a complex communication with references to multiple ontologies and/or rules involving various levels of Agents. *For example*, in the incident detection and response scenario (Setup 1) described in *section 8.2.1* and illustrated in *figure 8.6*, the ‘Inform’ message sent by the Grid Arbitrator Agent (*aAgtArb-6*) to the Central Control Arbitrator Agent (*aAgtArb-1*) started complex messages between multiple Agents at different levels using various ontologies and rules in order to generate a resolution plan to restore the traffic flow to normal.

The simulation results mentioned in *table 8.9* demonstrates the strength of the agent communication layer’s flexibility and functionality achieving more dynamic controllers Scope (as seen in *figure 8.10*) and quicker Resolution Time (RT) values (as seen in *figure 8.11*).

Communication overheads analysis

The implementation of SOA based systems can have some drawbacks mainly due to network latency, message payload overheads (increase data packets and processing) and services integration/discovery overheads (Erl et al., 2012). In highly distributed ITS environments, the services and host systems are geographically dispersed and network latency issues are bound to affect the performance especially where the bandwidth is low e.g. mobile/wireless communication (Schwab, 2007; Richardson & Ruby, 2007; Erl et al., 2012).

In this research, the SOA related overheads were addressed by considering different measures. First of all, the SOA Reference architecture of the overall ITS@CU platform (detailed in *chapter 7, section 7.2.3*) was streamlined to improve the integration, discovery and composition of the services layers and between the Agent controls. The agent’s communication performance and network latency issue were addressed by reducing the number of service calls/transactions between Agents at different levels (mentioned earlier in this section and *figure 8.13*) and also by reducing the web services data payload overheads and efficient XML processing.

Web service and XML overhead handling: The Agent Communication Layer is based on web services interface (mentioned in *chapter 7, section 7.3.1.2* and *figure 7.5*) which uses XML for the data payload. XML is highly flexible however it is not an efficient message format (Daigneau, 2011; Richardson & Ruby, 2007) due to parsing of elements tags and data. In a typical XML based web service the actual number of bytes required to construct a message is much higher than the actual information contained within. For example, a simple status/request message (similar to the example mentioned in *chapter 5, code excerpt 5.1*) with the size of the XML message wrapper of 832 bytes only has a message content size of 192 bytes which accounts to less than 25% of the size of the entire agent message.

Additionally, the web service connection setup and termination takes various steps. For example in the ITS@CU, a typical service call from a client Agent goes through the gateway middleware, service bus, application/web server hosting the Service, and then the database server and back to the client. Each step can result in web service failure hence the use of connection pooling, session management and message queuing is vital (*chapter 7* covers how ITS@CU address these issues at the implementation level).

In this research, the web service and XML related overheads were addressed firstly by reducing the payload size (by using compression techniques and avoiding SOAP for smaller transactions) and secondly by using efficient XML parsing to reduce processing overheads.

SOAP web services were all compressed using GZIP compression (using SOAP extension mentioned in *chapter 7*). As seen in the table 8.10, the GZIP compression reduced the data size and overall transaction download time of the web service requests by about 60% with an additional average 15-20% of time compressing and decompressing, yielding around 40-60% overall efficiency.

	Data Size (bytes) Simple SOAP Request Message	Transaction Time (ms) Including compression/ Decompression time	Data Size (bytes) Complex SOAP Message Response with Semantic-Content (Ontology/Rules)	Transaction Time (ms) Including compression/ Decompression time
Without Compression	832	13	17892	157
With GZIP Compression	278	7 <i>5 actual time + 2 decompression time</i>	6326	71 <i>62 actual time + 9 decompression time</i>

Table 8.10: SOAP Compression Results

Remarks: These values were captured using WireShark network protocol analysing software (wireshark.org). The web service transaction calls used in this study analysis were two types of messages 1) a simple Agent Request message representing basic status request (e.g. in *chapter 5, code excerpt 5.1*) and 2) a complex Response message with Ontologies and Rules (e.g. the plan implementation message by

Arbitrator agents in the resolve steps in the incident detection *scenario 8.2.1* and *figure 8.6*). These analyses were performed using the Control Agent application (mentioned in *chapter 7, section 7.3.2.4*) running on PDA emulation software for consistency and eliminating other network factors. The PDA emulators were hosted on a workstation PC with WireShark logging each message inbound/request and outbound/response with a detailed network analysis down to protocol and data packet level breakdown. The stop watch module embedded in the simulation applications was used for calculating the GZIP compression or decompression time values. The values in the result table *8.10*, *8.11* and *8.12* are the actual time of the message processing durations. Other duration values such as the pre-configured decision wait time values, connection setup and authentication/validation time were discarded by the stop watch module embedded in the simulation applications in order to analyse the actual duration relevant for analysing the results.

Although GZIP based compression in SOAP web services was helpful in significantly reducing the data payload, SOAP web services have a higher message overhead when compared to REST web services (Hameseder et al., 2011; Richardson & Ruby, 2007). So in order to further reduce the web services overheads, the Agent communication layer in ITS@CU supported both SOAP and REST based binding (mentioned in *chapter 7, section 7.3.1.2* and *figure 7.5*). For the purpose of performance comparison of the web service types, the same type of simulation tests mentioned in table 8.11 were performed using both SOAP and REST with XML as a data exchange format. The following table shows the results obtained:

	Data Size (bytes) Simple Request-Response Message	Transaction & processing Time (ms)	Data Size (bytes) Complex Message Request-Response with Semantic-Content (Ontology/Rules)	Transaction & processing Time (ms)
SOAP	278	7	6326	71
REST	113	5	3907	43

Table 8.11: SOAP and REST Comparison

Remark: During the test the Dynamic Compression feature for IIS7 was enabled for HTTPCompression.

As seen in table 8.11, the overall transaction and processing time using RESTful web services is more efficient when compared to SOAP web services, especially for a simple request-response service call. The reason for using SOAP along with REST web services in ITS@CU was to take advantage of both types. REST is very lightweight with less XML overheads (Erl et al., 2012; Richardson & Ruby, 2007) and therefore better suited for controllers. By contrast SOAP handles semantic content (as binary data) more efficiently, provides better type checking (Daigneau, 2011) and better implementation tools and support for .NET framework (Erl et al., 2010). It was therefore deemed efficient to use REST web services at the

controller level and SOAP web services for operational and service agents where the hosts generally have high bandwidth and CPU capability.

Although the use of compression techniques and the combination of RESTful and SOAP web services reduces the data payload, they both use XML for their data content. In order to reduce the XML processing overheads, the control applications utilise the latest .NET libraries for native parsing support, LINQ for in-memory XML document handling (DOM capabilities), and using XML data types in databases (SQL Server 2008) for quicker handling at the database level (Chapter 7 mentions the implementation details).

REST also provides support for other data exchange formats such as JSON and Fast InfoSet which are light weight as compared to XML (Hameseder et al., 2011; Daigneau, 2011), however XML was mainly adopted for the Agent Communication Layer due to its greater flexibility which far outweighs the lighter data payload benefit of JSON, Fast InfoSet or other object passing methods. Other benefits of using XML are mentioned in *chapter 5, section 5.2*.

Network latency considerations: In SOA based distributed environments, network latency and reliability is also a major issue especially in mobile/wireless scenarios (Schwab, 2007; Erl et al., 2012). In order to evaluate the Agents' communication approach using lower bandwidth and specification controller devices, a limited number of tests (similar as mentioned in *table 8.11*) were conducted using PDA devices using Wi-Fi and PDA emulators on a host PC.

Mode of traffic controller applications (<i>section 7.3.2.4</i>)	Data Size (<i>bytes</i>) REST based Status update Message	Average Transaction & processing Time (<i>ms</i>)	Data Size (<i>bytes</i>) SOAP based Message with Semantic-Content (<i>Ontology/Rules</i>)	Average Transaction & processing Time (<i>ms</i>)
PDA emulator	113	5	6326	71
PDA device	113	8	6326	128

Table 8.12: Network latency and CPU analysis (traffic controller application)

Modes:

PDA device (HTC P6500): CPU 400 MHz, 128 MB RAM with WiFi

PDA emulation mode (Visual Studio 2008 emulator): CPU 528 MHz, 512 MB RAM with stable host connection

Remark: PDA emulation software (Windows Mobile 6.5 Phone Emulator in Visual Studio 2008) was used to eliminate the hardware differences and intermittent communication factors.

The results in table 8.12 show the negative impact of lower bandwidth and CPU/Memory on the performance of Controller applications simulated using a PDA device and PDA emulator.

To conclude, agent's communication performance and network latency issues in the ITS@CU SOA platform were addressed by reducing the number of transactions between geographically distributed services/systems (controller to grid/central control level); reducing the payload size (by using compression techniques, and combination of REST and SOAP appropriately; and efficient XML parsing to reduce processing overheads. *Additionally*, various technologies and implementation methods adopted such as MSMQ for message queues, efficient session management and pooling implementation (using HttpSessionState in applications and IIS configuration on web servers) helped in improving the communication reliability in wireless scenarios.

Other findings and considerations

- Controllers per Control Agent ratio:** One of the key findings (*based on the MDDT result values mentioned in figure 8.8*) was that the ratio of the number Controllers devices controlled by a Control Agent is very important for overall communication performance. The larger the number of Controllers per Control Agent the lower is the need for outside communication (Between the Control Agent with other Agents). However the drawback is that a single Control Agent has to control the operation of multiple Controllers and work in a split-site hosting mechanism (described in *Chapter 4*) which results in extra intra-agent level communication (between Controllers/hosts) and can have a negative impact on the host controller resources. The best approach is to balance this ratio depending on the capacity, resource, communication technology and distance between the Controllers. In this study, the best ratio on average was 3 Controllers per Control Agent (PDA emulation mode). However, this can vary depending on the type and capacity of the Controllers, so this has to be approached based on network, CPU/memory and other such specification factors.
- XML Ontologies/Rules buildup:** A benefit of the Agent Communication Layer approach was that the ontologies and rules are exchanged on a need-to-know basis i.e. if an agent receives a domain ontology/rule set it is stored locally (in local database/files mentioned in chapter 5 and 7) and if a particular ontology is required to interpret an agent's message then the agent acquires it using a request message (mentioned in *chapter 6*). This avoids the need of embedding ontologies/rules XML data within agent transactions (semantic-content) and the local ontology and rules build-up overtime hence reducing download data size over time.

- **XML Message splitting:** The Agent communication layer also supports message splitting using the dynamic XML node distribution mechanism in the message assembler/parser module. In this way, an Agent need send only the relevant parts (XML Node) of the message and domain-level ontologies to multiple Agents (mentioned in chapter 5). This mechanism can also reduce the communication overheads and improves the integration between different domain controls.

8.2.3. Semantic web services approach

The Semantic Web Services concept in the platform's Agent communication implementation was evaluated to assess the following key benefits:

Dynamic Service discovery: The Service Agents in the ITS@CU platform were designed to dynamically build its metadata (called meta-tags) based on its domain-level ontologies and capabilities/operations. This helps other Agents in finding the most suitable Service Agent (over the platform Service bus) for any service delivery related requests.

This functionally was evaluated in simulation study (using setup 1) but with 3 Weather Info Service Agents (*oAgtSvcWS-n*) with each Service Agent responsible for a specific set of Grids. First, various simulated web service requests based on different ontologies were made over a period (40 queries for 10 minutes) to automatically generate meta-tags for each Service Agent's operations (web service methods). Then, to assess the outcome, different Grid Arbitrator Agents (*oAgtArb-n*) were programmed to find a Service Agent which had the most recent weather update for a specific link. For example (using the Agents described in *section 8.1* and study area *figure 8.1*) when the Grid Arbitrator Agent for *Grid-6* enquired the Service registry Agent (*oAgtSvcReg-1*) for a service lookup request regarding weather update relevant to "Foleshill Road" (incident location point in *figure 8.3*), the *oAgtSvcReg-1* used the meta-tags and identified the *sAgtSvcWS-1* as suitable to fulfil the request as it had a meta-tag "Foleshill road" as a link name. This meta-tag was added when the *sAgtSvcWS-1* first received the domain-level ontology (during the simulated web service requests) where the link names list also contained "Foleshill road". In this way meta-tags are built up dynamically for each service Agent and if multiple service agents have the same meta-tags then *oAgtSvcReg-1* will look for other meta-tags until a service Agent is found with the most matching meta-tags.

Semantic queries (semantic layer): The semantic web services capability in the platforms' Agent Communication Layer/Interface (described in *chapter 5, section 5.1* and *chapter 7, section 7.3*) allows the augmentation of semantic-content (described in *chapter 6*) and within Agent messages. The semantic-content provides the flexibility to embed complex queries and conversations which the targeted Control Agents can interpret based on its domain knowledge (domain-level ontologies) and can delegate/forward/coordinate the message to other Agents.

In order to evaluate the semantic layer, consider the “resolve” step in the detection and response scenario (described in *section 8.2.1* and *figure 8.6*) where the Central Control Arbitrator Agent (*oAgtArb-1*) delegated the Plan implementation request to Grid-Level Arbitrator Agents (*oAgtArb-6* and *oAgtArb-8*) using the semantic-content, only mentioning the potential diversion routes but with no detailed actions. The receiving agents further elaborated the Plan using their local domain-level Ontologies and Rules, and then further delegated these tasks to the relevant Controller-Level Agents (*cAgtCtrTL-n* and *cAgtCtrMsgD-n* seen in *figure 8.6*). Each *cAgtCtrMsgD-n* and *cAgtCtrTL-n* Agents interprets the semantic-content in the message instructions (based on their local ontologies and Rules) in order to execute the actions (mentioned in the *figure 8.6* and its description) for the restoration of the traffic flow rate.

8.3. General observations & discussion

8.3.1. Reliability

The following are some of the reliability related aspects of the platform relevant to the Agent-based Controls approach:

- A single Agent can deal with multiple requests at the same time however this also means a single point of failure in the platform in case the Agent becomes overloaded or becomes non-functional. So for the purpose of resiliency and load balancing, a pool (pair or a cluster) of Agents can be configured in the platform which work together and provide similar role functions to a single agent (mentioned in *chapter 4, section 4.14*). In such cases one Agent within the pool becomes a pool manager to distribute requests evenly. The pooling mechanism is particularly useful for agents with vital functions, so if one of the vital agents (or its host) stops responding or becomes overloaded then the other Agent(s) in that pool takes over and the platform continues to function.

This functionality was used in incident detection and response scenario, where the Arbitrator Agents at the central control-level (*oAgtArb-1* and *oAgtArb-2*) shared the tasks, and the *oAgtArb-1* served as the pool manager during the simulations (see *figure 8.2* and *table 8.4*).

- The ITS@CU platform has QoS Operational Agents (*oAgtQoS-n*) which provides the current performance and operational status of the platform's agents in terms of their Quality of Service (QoS) values 3-0 (described in *chapter 4, table 4.7*). The *oAgtQoS-n* is used by other Agents to aid in selecting a suitable Agent to carry out task(s) where multiple Agents with similar capabilities are available. This is particularly useful in selecting Service agents or Arbitrator Agents which may be overloaded or slow responding due to other agents' requests.

The QoS agents improves reliability by keeping track of agents current status but also by informing relevant Manager Agents if an Agent has 0 or 1 value, so an appropriate action (e.g. create new instance) can be taken by the relevant Manager Agent. Although the QoS agent adds an additional communication overhead (due to the periodic ping broadcast messages to agents) it is a relatively small overhead when compared to the benefits it provides. The simulation results shown in *figure 8.13* show that the system (with a ping broadcast interval of 3 minutes) performs better (with a lower number of transactions) than the setup without agent-based controls.

The interval must be configured based on the number of agents, the network and other such factors to avoid any adverse impact on the overall performance of the system due to disproportionate QoS value check transactions. Additionally, the ping interval configuration implemented in the platform was not configurable for individual agents (i.e. the interval value was same for all agents). As a future recommendation, individual agent's status/health check can be more efficient as compared to broadcast so a crucial agent can be checked more often than less important agents to keep the number of transactions lower.

- The agent communication layer in the platform includes a message type “Reply” with instruction-types *Acknowledgment* and *Failure* which were especially implemented for a reliable delivery of agent messages and handling message corruption issues (see *chapter 5, table 5.1*). By default every message sent by an Agent is acknowledged by the other agent, and if an acknowledgement reply is not received after a set period of time (10 seconds used in the simulations) then the Agent retries or takes another action based on the configuration. If the message is received but not understood (usually due to ambiguous ontology/rules or message corruption during the multi agent transaction) a *Failure* reply message with reason code is sent back to the sender(s).

Additionally, from the technology point of view MSMQ was used in the platform to queue agent messages by the sender (host Controller) so in the event of connection loss, the data is queued locally and as soon as the connection is re-established the data is sent in an orderly fashion. This is particularly useful in wireless/mobile connectivity where the communication channels are not reliable.

- The controller applications (mentioned in chapter 7, section 7.3.2.4) were designed to switch to default-logic mode (i.e. the basic default functionality/logic of a controller). So if a control agent (Agent-Logic functionality) stops working due to connectivity failure or control agent malfunction then the traffic Controller continues its basic default functions in a controlled manner despite the loss of communication with its control agent. It must be considered by any such critical agent based controls system to ensure that the system does not become non-operational during such incidents/issues.
- The split-site agent hosting mechanism (described in *chapter 4, section 4.1.4*) allows the Agent to efficiently use the host resources by distributing its operations and files on multiple host Controllers. In this way, Agent-based Controls can actually perform more functions on the same hardware. Although the split-site hosting mechanism has various performance and recourse sharing advantages, it could also impact the agent's reliability if any of the involved hosts become unresponsive or slow. This is especially true if the primary host fails as the Agent also fails despite

the other partial hosts being operational. It is therefore recommended to use the split site hosting mechanism only when the participating hosts have reasonable capacity/recourses available for the agent's functions and also by considering network capacity and link type/channel between the participating hosts to avoid any impact on the overall system reliability.

8.3.2. Security handing

The security in Semantic Agent-based Controls is of significance importance as any vulnerability can impact the entire platform. The security was therefore designed as part of the Agent Communication Layer/Interface (described in *chapter 5*) which requires the transactions authentication by a Security Agent which also validates the identity of the sender Agent and its level of Access. As far as the technical implementation was concerned, all the web services based transactions in the platform were encrypted using AES-128 encryption in order to secure the transport method and the communication channel (Wi-Fi or 3G/GPRS).

The security was assessed by analysing various test cases such as creating an agent with less privileges/access level (e.g. the grid Arbitrator Agent *oAgtArb-6* in incident detection and response scenario) then using that agent to try and obtain ontologies allowed only for the Central control-level Arbitrator Agent *oAgtArb-1*. The transaction resulted in access being denied. Similarly an unauthorised Agent-ID was used and similar results were obtained.

In another test scenario, a Controllers-Level Agent was deliberately altered during simulations by changing its value at the database level to simulate situations where an Agent (with low access level i.e. Controllers-Level) was trying to perform operations at the Grid-Level. The Security Agent was successful in identifying the Agent's actual access level (due to the ID cross checking in the main database functionality implemented in the platform) and changed its state to "suspended".

These studies were conducted to demonstrate that the ITS@CU approach of using Agents is secure and most importantly to demonstrate that Agent-based Controls can be securely used in any such environment as long as the right security measures are undertaken (such as those suggested in *chapter 5, section 5.4*).

8.3.3. Centralisation and decentralisation flexibility

The Agent-based Controls approach promotes de-centralisation due to its localised decision-making capability at the Controllers-Level. If the controllers are in default mode (i.e. simulated as fixed controllers) the system becomes more centrally controlled as each controller would require direct communication with a grid control or the central control system. However, the Agent-based Controls coordinate with each other and perform decision-making at the Controller-Level (as seen in *figure 8.2*) with fewer interactions with Grid-Level Agents except for Arbitration purposes. For example, in the incident detection scenario as demonstrated in *figure 8.6*, the incident was reported to Arbitrator Agent (*aAgtArb-6*) at the Grid-Level only when the incident location was identified and confirmed first by the Control Agents (*cAgtCtrVC-n* and *cAgtCtrLnk-n*).

In the ITS@CU platform, one of the main advantages of the Agent-based Controls approach was the flexibility of adjusting the level of centralisation and decentralisation of the system. The communication and relationship between all the entities (Agents, Controllers etc.) were all based on the ontologies (as described in *chapter 5* and *6*), and due to the flexible design of these ontologies the system can be configured to any level of centralisation/decentralisation. The more comprehensive the relationship ontologies (domain-level) with associated select and match Rules are defined, the more they would enable decision making abilities at the Controllers-Level. Similarly these Ontologies and Rules can be designed in such a way that will designate all the decision-making to the Grid-Level or even to the Central Controller-level.

8.3.4. Maintenance and scalability

The following are some of the key maintenance and scalability aspects of the platform relevant to the Agent-based Controls approach:

- One of the other important advantages of the Agent-Based Controls method is its dynamic update capability without the need to change the hardware or even firmware. The functionality of a Controller in the platform can be updated by sending new Agent-Logic operations/commands embedded in a request message. This functionality was evaluated by updating the Agent-Logic of all the 36 Traffic Light Controllers (*CtrTL-n* with 182 associated traffic light controls) mentioned in *table 8.1*. All the controllers were successfully updated in less than 2 minutes (using PDA emulators and just adding an extra operation to the Agent-Logic). This functionally cannot be achieved in a traditional control system as most of the current systems require hardware or

firmware refresh which is costly and could involve more Controller down time which is not ideal for a traffic system affecting the road users.

- Similar to the incident detection behaviour of the Control Agents (described in detection and response scenario above), the Control Agents can also detect any hardware malfunctions/defects. Each Agent constantly monitors the host Controller and its associated traffic Controls, and if a problem is noticed it raises an Inform message to a Control Agents Manager (*oAgtC_AgtMgr*) with the relevant defect code. Similarly, if the Control Agent stops responding (even if its host Controller is functioning correctly), the Controllers switch to default-mode and the Control Agents Manager notifies other relevant Agents (about the non-functioning Control Agent/Controller). During the simulation studies, the PDA emulators were deliberately switched-off and restarted to simulate Agents and Controllers having a temporary defect or permanent failure. The Control Agents Manager successfully raised Agent and Controller failure events with defect codes during all these tests.
- Another major advantage of this approach is the ability to define different Ontologies and Rules based on a location's requirements. The ontologies and rules can be adapted for a specific section of the traffic network (for example city center or even smaller sections relevant to specific places such as schools) in order to maximise the overall performance of traffic Controllers and update the functionality over the time specific to that location.

8.3.5. Limitations

The semantic-agent based control approach has the following limitations:

- Although the ontologies and rules provide a high level of flexibility, they need to be defined in an optimised manner to avoid any negative impact on the operations of the agent-based controls. For example poorly defined rules can lead to ineffective decision making by Control Agents, similarly a lack of ontologies or ambiguous entities within the ontologies' descriptions can also limit the decision making of the Control Agents. The top-level ontology for the ITS@CU platform mentioned in *chapter 6, figure 6.5* and in "*Appendix O*" is an example of an optimised ontology with no ambiguity.

- The semantic agent-based controls approach requires controllers to be interconnected and capable of hosting the control agents. Modern traffic control systems are heading in this direction however the legacy infrastructures may not be capable of implementing this approach.
- The evaluation studies involved a single region and 6 grids in an urban traffic network. Multiple regions and other transportation types for example rail/ferry network were not evaluated due to the specific requirement and data availability from Coventry City Council.
- The ITS@CU platform works in a pre-configured manner and the simulation/demonstration requires the specific data format and hardware/software environment pre-requisites.
- The agent communication layer in the ITS@CU platform supports SOAP, WCF and RESTful web services (as described in *chapter 7, figure 7.5*) so the participating components/systems in the proposed approach must conform to SOA principles and must have the capability of using services based communication. The transportation industry is generally moving towards adopting SOA enabled platforms and technologies (Wang et al., 2010; Dion Hinchcliffe, 2009; Ross Altman, 2008), which are capable of supporting this approach.

8.4. Conclusion and summary of findings

This chapter evaluated the overall research approach of using the novel concepts of Semantic Agent-based Controls. The results from the study cases demonstrated the following key advantages and findings of the approach:

- Semantic agent-based controls demonstrated the capability to detect problems such as incidents and hardware issues more efficiently, as seen in *figure 8.9* showing a high detection rate due to the low number of false alarms. The key advantage of the agent-based controls is their ability to respond to such problems by coordinating with each other and making decisions as demonstrated in *figure 8.11 and 8.12* showing a quicker restoration time by using controllers' scope more effectively. Traditional fixed controllers (for example SCOOT system) are generally centralised and only react in a pre-configured manner with a fixed scope hence they do not have the dynamic self-organisation capabilities of the agent-based controls approach.
- The agent-based controls approach demonstrated a significant reduction in the number of transactions between the traffic controllers and central/grid control systems (as seen in *figure 8.13*). This is due to control agents communicating on behalf of their host controllers primarily at the controllers-level between agents in the same grid and only communicating essential information up to the grid level agents.
- The Agent communication layer provides a high-level of flexibility in the form of semantic-content embedded in Agent messages allowing complex queries and conversations which the targeted Control Agents can interpret based on their domain knowledge (domain-level ontologies) and can delegate/forward/coordinate the messages to other Agents.
- The semantic layer helped Agents in finding the most suitable Service Agent (over the platform service bus/layer, as described in *chapter 7, section 7.3.1.2*) for any service delivery related requests using the meta-tags which are updated automatically each time an agent processes a new/update ontology.
- The features and mechanisms such as the split-site hosting, QoS ratings, and load balancing employed in this research helped improve the reliability of Agents and provided a dynamic performance management capability as demonstrated by the pooling mechanism which enables the agents to work in clusters/pools to share the load and provide resiliency in case of an Agent failure. Additionally, the switching capability of the agent-based controls approach (default-logic mode) ensures that the Controllers continue their important operations in case of Agent failures.

- The security measures for the semantic agent-based controls approach must incorporate identity authentication and level of access validation as a minimum due to their importance in securing the platform as demonstrated in *section 8.2.6*. Additionally, the communication/transmission channels must also be secured.
- The semantic agent-based controls approach presented in the research promotes decentralisation due to its localised decision-making capability at the controllers-level as demonstrated in the incident detection scenario. The advantage of the approach is the flexibility to adjust the level of platform's centralisation/decentralisation by defining the ontologies and rules accordingly. The communication and relationship between all the entities (Agents, Controllers etc.) were all based on the ontologies, and due to the flexible design of these ontologies the system can be configured to any level of centralisation/decentralisation i.e. the more comprehensive the relationship ontologies (domain-level) with associated select and match Rules are defined, the more they would enable decision making abilities at the Controllers-Level. Similarly these Ontologies and Rules can be designed in such a way that will designate all the decision-making to the Grid-Level or even to the Central Controller-level.
- Semantic Agent-based Controls are ideal for ITS as they provide an efficient way of updating and optimising the traffic Controllers/Controls capabilities without the need to change the hardware or even firmware in form of Agent-Logic update. It is a cost effective way to develop, maintain and expand/upgrade a transportation system in a future connected environments.

Chapter 9

Conclusion & Further Work

The previous chapter evaluated the overall research approach of Semantic Agent-based Controls in a SOA based ITS environment. It presented an analysis of the ITS@CU platform by simulating different traffic scenarios and test cases.

This chapter concludes the thesis, summarises and highlights the significance of the findings. It presents suggestions for further work and the potential of the research approach in different application areas.

9.1. Conclusion

The research presented in this thesis was aimed at addressing the challenges of integration, communication and management which are associated with large scale multi-domain systems due to their highly complex, distributed and multi-domain natures. The application domain selected to evaluate the novel implementation approach of semantic agent-based controls was Intelligent Transportation System (ITS). ITS@CU, a comprehensive, purpose built, multi agent simulation and development platform was developed to investigate urban traffic control issues. A number of commercially oriented applications were also developed and have since been successfully deployed.

The software designs and approach reported in this research addressed the challenges offered by the complexity, distributed and multi-domain nature of ITS in the following ways:

- 9 types of Control Agents with different roles were designed to govern the functionality of multiple traffic controllers. The control agents were based on intelligent and mobile agents design properties. These control agents host their Agent-Logic and associated knowledge base across a set of specific interconnected Controllers (for example traffic lights controller units).
- The control agents were supported by various types of Operational and Services Agents in the multi-agent platform. Operational agents provide key supporting functions such as arbitration between agents, security, and platform/agents management and performance monitoring. Service agents provide access to external services by different sources for example direction/route services provided by Microsoft Bing Maps and weather information provided by the Met Office. The close cooperation and coordination between all the agents having specific roles provided the Agent-based Controls with the mechanism required to cooperatively respond to emergent traffic situations.
- Service Oriented Architecture (SOA) principles provided seamless integration and communication methodologies in the form of standard services based interfaces which was required for the interoperability of the technologically diverse and multi-domain nature of the traffic systems/devices.
- A unique agent communication layer was developed to allow the efficient communication and coordination mechanism between the agents across the SOA enabled platform components. The agent communication messages (implemented as web service operations) comprised of message instructions and a semantic content layer. The message instruction commands were adapted from FIPA standards but customised for better performance in the agent communication layer implementation.

- The semantic content layer within agent messages provided a sophisticated and flexible method to embed domain specific commands with associated ontologies and rules that semantically describe the Agents' intentions. This enables agents to coordinate cooperatively across the SOA enabled network of traffic systems and devices, and formulate action plans in the form of decision rules between the semantic agent-based controls facilitating dynamic and distributed decision making in response to emergent situations. Overall, it facilitated efficient detection of traffic control problems (congestion/incidents/system faults) and the making of intelligent/smart decisions by activating/adapting different traffic controls.

The ITS@CU platform was developed from the ground up specifically to support and evaluate the implementation approach presented in the Thesis. The platform and the associated simulation utilities were developed using the latest state of the art commercially available tools and technologies such as .NET 3.5/4, WCF, ASP.NET, SQL Server 2008 spatial data, LINQ, Microsoft Silverlight, Bing Maps API, Windows Mobile 6.5 SDK and various wireless communication technologies and libraries.

Research findings

The research approaches were evaluated using a variety of different test cases generated using historical traffic data and typical traffic situations identified by UTMC Control room, Coventry city council. The results demonstrated the advantages of semantic agent-based controls especially when compared with non-agent based controls, where applicable.

The following are key advantages and findings of the overall research approach evaluation:

- Semantic agent-based controls demonstrated the capability to detect problems such as incidents and hardware issues more efficiently. The detection rate was improved by 10-30% with a combined reduction in the number of false alarms. The key advantage of the agent-based controls is their ability to respond to such problems by coordinating with each other and making decisions. The overall traffic flow restoration time was improved by approximately 20%. Traditional fixed controllers are generally centralised and only react in a pre-configured manner with a fixed scope hence they do not have the dynamic self-organisation capabilities of the agent-based controls approach.
- The agent-based controls approach demonstrated a significant reduction (approximately 75%) in the number of transactions between the traffic controllers and central/grid control systems when compared with similar scenarios run with the controllers in default mode. This is due to control

agents communicating on behalf of their host controllers primarily at the controllers-level between agents in the same grid and only communicating essential information with the grid level agents.

- The Agent communication layer provides a high-level of flexibility in the form of semantic-content embedded in Agent messages allowing complex queries and conversations which the targeted Control Agents can interpret based on their domain knowledge (domain-level ontologies) and can delegate/forward/coordinate the messages to other Agents.
- The semantic layer helped Agents in finding the most suitable Service Agent (over the platform service bus/layer) for any service delivery related requests using the meta-tags which are updated automatically each time an agent processes a new/updated ontology.
- The features and mechanisms such as the split-site hosting, QoS ratings, and load balancing employed in this research helped improve the reliability of Agents and provided a dynamic performance management capability as demonstrated by the pooling mechanism which enables the agents to work in clusters/pools to share the load and provide resiliency in case of an Agent failure. Additionally, the switching capability of the agent-based controls approach (default-logic mode) ensures that the traffic Controllers continue their important operations in case of Agent failures.
- The semantic agent-based controls approach presented in the research promotes decentralisation due to its localised decision-making capability at the controllers-level as demonstrated in the evaluation studies in incident detection scenario. The advantage of the approach is the flexibility to adjust the level of the platform's centralisation/decentralisation by defining the ontologies and rules accordingly. Similarly, the level of ontologies and rules can be adjusted specifically for regions allowing optimisation based on the specific requirements of the location/traffic section or grid.
- The security measures for the semantic agent-based controls approach must incorporate identity authentication and level of access validation as a minimum due to their importance in securing the platform. Additionally, the communication/transmission channels must also be secured.
- Semantic Agent-based Controls are ideal for ITS as they provide an efficient way of updating and optimising the traffic Controllers/Controls capabilities without the need to change the hardware or even firmware in the form of Agent-Logic updates. It is a cost effective way to develop, maintain and expand/upgrade a transportation system in a future connected environment.

Critical evaluation of the research contributions and novelty

The main novelty in terms of the implementation approach is the introduction of the concept of “Semantic Agent-based Controls”. It enables the controlling of multiple ITS-based controls/systems over a network by using Control Agents. The thesis also introduces a novel “agent communication layer” which provides the capability to encapsulate sophisticated semantic data (commands, ontologies and action rules) within agent messages. This enables efficient communication, cooperation and coordination between the semantic agent-based controls, and provides the ability for agents to work in synergy with other agents with dynamic adaptability in order to manage the traffic grids’ controls and make smart decisions in response to emergent situations.

The second major contribution is the incorporation of the agent communication layer within the SOA enabled ITS@CU platform to facilitate Agent interactions in the form of services over highly complex, distributed and multi-domain networks. This is a unique implementation of multi-agent systems and provides a robust and highly scalable platform which can be also adopted in a wide range of other application domains.

The main novelty in terms of application is the new ITS platform (ITS@CU) which provides comprehensive agent management, communication and traffic control devices/systems simulation capabilities using the adopted implementation approaches and technologies. The ITS@CU platform as a whole is unique and no such platform currently exists which uses .NET technologies and the agent oriented approach adopted in this research for ITS.

One of the most significant contributions in terms of technology enablers is the Mobile Application Development Framework (MADF). It provides a template based rapid mobile application development framework (in the form of project templates in Visual Studio 2008) with reusable classes, modules and tools specifically designed to simulate the traffic controllers and controls behaviour. MADF was used for developing the key simulation applications (such as traffic controllers, controls and vehicle controls) on PDA devices for the evaluation of the ITS@CU platform. Additionally, it provided a baseline framework which was adapted to develop some of the commercial applications during the project such as the Patient Transportation System (PTS) and the Intelligent Vehicle tracking solution.

Additional contributions

The research project had a commercially oriented R&D focus. In addition to the core research objectives, elements of the ITS@CU platform developed during the project also led to the development of various proof of concepts and commercial applications/products for T@lecom. The following are three successful applications which resulted from this research:

- Patient Transport System (PTS):** It is a mobile solution for non-emergency ambulance crews to transfer patients to and from hospitals. The solution consists of a mobile/PDA application integrated with NHS control room system via the T@lecom mobile gateway. The key relevant features include intelligent tracking/routing, two-way messaging/alerts and patient prioritising in real-time. The main contributions of the research were the use of MADF modules and class libraries in the development of the PTS mobile application.
- Mobile Gateway System:** It is a SOA enabled unified communication and integration platform for T@lecom's mobile solutions. It comprises of a set of web services and a web-based management application for managing and monitoring mobile/PDA applications and devices. The main contribution of the research was the SOA based communication layer developed for ITS@CU and adapted for this system with re-use of various services and modules/classes. As a result of this contribution, T@lecom's gateway was improved with the ability to integrate WCF enabled mobile applications with the existing gateway and better service choreography/services composition for streamlining mobile processes/work flows.
- Vehicle Tracking Solution:** It is an asset tracking solution for different type of assets such as vehicles and mobile devices. It comprises of a mobile/PDA application and a web based monitoring and management application. The main features include geo-fencing based tracking and alert notifications, real-time driver/vehicle updates, remote driving and navigation instructions and a novel feature of driver behaviour assessment. It incorporated various modules and functionalities originally developed for the ITS@CU platform's simulation/monitoring application.

Additionally, various other relevant research applications such as a Bluetooth based Vehicle-to-Vehicle communication application with ad-hoc wireless networking capability and a GPS based incident detection system were developed using the concepts/modules from this research activities.

Further details regarding the above mentioned applications and contributions is provided in "*Appendix A* and *B*."

9.2. Other potential application areas (Non-ITS)

The implementation approach of semantic agent-based controls presented in this research is not limited to ITS. It offers an interesting potential in numerous application areas. Some of the areas identified include:

City infrastructure management: Similar to traffic management systems, modern cities involve a range of systems and services distributed in multi-domain environments. The concept of smart cities, which requires all these systems and services to be fully integrated and work in a collaborative manner, seems to be an ideal candidate for adopting the semantic agent-based controls approach. This approach can provide such levels of integration and collaboration between the systems and services due to its SOA enabled flexible semantic communication layer. It can provide enhanced and dynamic capabilities using control agents with location-aware decision making. Other benefits of the approach in this application area are its support for grid based distribution of host system resources and its capability to integrate with any SOA enabled infrastructure including cloud based services.

Utility services/infrastructure management: This implementation approach can be applied to the utility sector such as telephone/broadband or electricity/gas supply. Their geographically distributed hardware/devices and services can be managed in a more effective manner using the semantic agent-based controls. In case of faults and system load issues, control agents can respond much more effectively without the need for a highly centralised monitoring system. This will help in keeping the services operational resulting in good customer experiences. It can be cost effective due to reduced maintenance issues by allowing self-organisation and dynamic update capabilities in Agent-based controls (as demonstrated in the thesis evaluation chapter).

Healthcare systems: Modern hospitals and associated care providers presents a good application area where hospitals' systems can be well-integrated and managed via the semantic agent-based controls approach. For example, the equipment/systems in patient rooms/wards, mobile patient health monitoring devices, doctors/nurses portable tablet devices can be all linked and managed by the semantic agent-based controls, which can provide proactive patient monitoring by means of controls agents cooperatively managing the involved systems and alerting relevant staff quicker as well as responding to emergencies in a controlled manner. Such control can be achieved by adjusting the rules and ontology levels which is another advantage of this approach.

Marketing campaigns: This is another interesting application area where digital interactive displays/billboards across shopping centres can be dynamically controlled and updated by using the semantic agent-based controls approach. It can be used for location and situation aware advertisement displays where, for example, holidays or even sudden weather changes can attract a different type of consumer. The system could be implemented by integrating the control agents (controlling distributed digital interactive

displays/billboards) with external services for dynamic updates based on the location aware domain ontologies.

Business process monitoring: Modern businesses involve multiple business processes in the form of services usually geographically distributed across multiple branches and departments. The Service Agents concept in this approach can be adapted to monitor and control these business process services across SOA enabled environments (as demonstrated in the Thesis, service agent controlling and accessing external services such as Bing Maps and the weather services). In such environments, service agents can effectively monitor the services, help in service discovery using the meta-tags, load sharing/balance using the agent pooling mechanism and enable the system to dynamically adapt to the failure of a single service and can therefore be much more robust and reliable.

9.3. Further Work

During the course of this research the following areas of further work have been identified:

- The ITS@CU platform was evaluated using one urban region with six grids. The scope of the traffic network area can be further increased by designing and adding more domain-level ontologies to describe wider transportation infrastructure elements and relationship. Similarly, more accompanying rules can be implemented at controllers-level to further devolve the distributed decision making. This would enable the evaluation of the adopted approach for different types and levels of transportations with multiple regions including rail and ferries network.
- The ITS@CU platform can be extended to include vehicles to be controlled by Control Agents, where each vehicle can be dynamically controlled by vehicle control agents in different grids. It will take the research to a next level where Control Agents for vehicles can work in synergy allowing the development of an advance vehicle mesh network. This can be even approached using the vehicle-2-vehicle ad-hoc wireless networks. In this way vehicles can cooperate with each other by passing key information (such as congestion ahead, speed warning and other such alerts/messages) and also communicate with grid/infrastructure controls using the agent communication layer presented in this research. Such implementation can lead towards the realisation of smart connected vehicle on smart roads.
- The platform can be incorporated with cloud based services particularly the federated form of cloud computing. Service Agents presented in this research can be modified to provide cloud

services to the ITS@CU platform including shared data repository and even processing in the cloud for CPU intensive agent tasks. This will further reduce the need for traffic controllers with high processing power as agents decision type tasks can be performed in the cloud. However, high speed and reliable connectivity is required in that case.

- The signal adaptation in the research was limited to severe congestion response only. The adaptation process can be further extended to mild congestions or even normal traffic conditions to increase the overall traffic flow. The Traffic Lights Control Agents, Vehicle Count Sensors Control Agents and Link Control Agents can be configured to cooperatively decide the level of traffic light adjustment (reducing/increasing green time) allowing dynamic adaptation in more effective manner (involving multiple links and routes) as compared to traditional fixed systems for example SCOOT which perform signal adaptation in a pre-configured manner and only at the junction levels.
- The platform capabilities can be extended further to evaluate the effectiveness of dynamic sign boards which have the potential to replace fixed signs in future ITS systems. In ITS@CU, the dynamic sign boards can be controlled by Control Agent in a similar way as other controllers and they can be used intelligently to display different traffic signs based on situations, location and time (for example the speed limits of signs on different roads can be increased or decreased, directions can be set to one-way/two-way and stop/no stop signs can be changed and so on).

T@lecom's research and development activities have been capitalising on the foundation of the existing platform and its associated utilities, components and modules. T@lecom intends to advance and improve the elements of the ITS@CU system particularly MADF to provide more flexibility and functionality to their clients.

The research at CTAC will continue and plans are already under way to extend this research and make the "non-commercial" elements of the platform available for further research.

Additionally, some of the commercially insensitive elements of the ITS@CU platform can be modularised, packaged up with relevant Application Programming Interface (API) documentation and shared with the open source community in order to promote use of the platform in similar studies.

References

- Ali Arsanjani, 2004. *Service-oriented modeling and architecture, How to identify, specify, and realize services for your SOA*. [Online] Available at: <http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/> [Accessed 2010 June 3].
- Altman, R., 2008. *The Future of SOA -- Myths and Realities*, <http://www.soa-consortium.org/podcasts-webcasts/Santa-Clara-08/podcast-ra.htm> [Accessed 13 July 2009].
- Altova, 2007. *What is the Semantic Web?* [Online] Altova Available at: http://www.altova.com/semantic_web.html [Accessed 12 April 2010].
- Amatriain, X., 2008. *Agile Research*. [Online] Available at: <http://technocalifornia.blogspot.com/2008/06/agile-research.html> [Accessed 16 April 2012].
- Ambler, S.W., 2008. *Agile Adoption Rate Survey Results: February 2008*. [Online] Available at: <http://www.ambysoft.com/surveys/agileFebruary2008.html> [Accessed 20 April 2012].
- Andreas L. Symeonidis and Pericles A. Mitkas, 2005. *Agent Intelligence through Data Mining*, Springer US, pp. 41-45.
- Anon., 2009. *Semantic Agents on the Internet - Ontologies for Internet Agents, Search Agents*. [Online] Available at: <http://encyclopedia.jrank.org/articles/pages/6891/Semantic-Agents-on-the-Internet.html> [Accessed 19 May 2009].
- Antoniou, G. & Harmelen, F.v., 2009. *Web Ontology Language: OWL*. pp.91-110.
- APMG, 2011. *PRojects IN Controlled Environments*. [Online] Available at: <http://www.prince-officialsite.com/> [Accessed 18 April 2012].
- Balan, G. & Luke, S., 2006. *History-based traffic control*, 2006. ACM.
- Balbo, F. & Pinson, S., 2005. *Dynamic modeling of a disturbance in a multi-agent system for traffic regulation*. *Decision Support Syst.*, 41(1), pp.131-146.
- Basnayake, C., 2004. *Automated traffic incident detection with equipped probe vehicles*. Proceedings of the 17th International Technical Meeting of the Satellite Division of the Institute of Navigation, 741-750
- Bauer, B., Müller, J.P. & Odell, J., 2001. *Agent UML: A Formalism for Specifying Multiagent Interaction*. *Agent-Oriented Software Engineering*, pp.91-103.

- Bazghandi, A. & Pouyan, A.A., 2011. *An Agent-Based Simulation Model for Urban Traffic System*. Computer and Information Science, 4(4).
- Bazzan, C., Wahle, J. and Klügl F., 1999. *Agents in Traffic Modelling - From Reactive to Social Behaviour*. Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 1701.
- Bellifemine, F, Caire, G, Greenwood, D, 2007. *Developing Multi-Agent Systems with JADE*, John Wiley & Sons, Ltd
- Booth, D., Haas, H., F. McCabe, Newcomer, E. , Champion, M. , Ferris, C., and Orchard, D., 2004. *Web services architecture*. Note 11, W3C Working Group.
- Borselius, N., 2003. *Mobile agent security*. Electronics & Communication Engineering Journal, 14(5), pp. 211-218.
- Bradshaw, J., 1997. *Software Agents*, MIT Press, Cambridge, MA.
- Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N. R. and Treur, J. (1997) *DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework*. Int. Journal of Cooperative Information Systems, 6, (1), 67-94
- Buchanan, W.J., Naylor, M. & Scott, A.V., 2000. *Enhancing network management using mobile agents*. In Seventh IEEE International Conference and Workshop on the Engineering of Computer Based Systems., 2000.
- Busetta, P., Bailey, J. & Ramamohanarao, K., 2003. *A Reliable Computational Model For BDI Agents*.
- Cardosa, J., 2007. *The semantic web vision: where we are?*. IEEE Intelligent Systems 22(5), pp.84 -88.
- Chen, B. & Cheng, H.H., 2010. *A Review of the Applications of Agent Technology in Traffic and Transportation Systems*. Intelligent Transportation Systems, IEEE Transactions on, , 11, pp.485 -497.
- Chen, B., Cheng, H.H. & Palen, J., 2006. *Mobile-C: a mobile agent platform for mobile C-C++ agents*. Softw. Pract. Exper., 36(15), pp.1711–1733.
- Chen, B., Linz, D.D. & Cheng, H.H., 2008. *XML-based agent communication, migration and computation in mobile agent systems*. J. Syst. Softw., 81(8), pp.1364--1376.
- Chowdhury, M.A. & Sadek, A.W., 2003. *Fundamentals of Intelligent Transportation Systems Planning*. Artech House.
- Clemens Portele, 2007. *OpenGIS Geography Markup Language (GML) Encoding Standard*. [Online] Available at: https://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/%3fartifact_id=20509 [Accessed 12 September 2009].

- Clement, S. & Taylor, M., 2006. The simple platoon advancement model of its technologies applied to vehicle control at signalised intersections. *Journal of Advanced Transportation*, 40, pp.1-21.
- Collaris, R.-A. & Dekker, , 2010. Scrum and RUP - A Comparison Doesn't Go on All Fours. *Agile Record*, (1), pp.62-65.
- Cubillos, C, Guidi-Polanco, F. & Demartini, C., 2005. MADARP: Multi-Agent Architecture for Passenger Transportation Systems. *Proceedings of the 8th IEEE International Conference on Intelligent Transportation Systems (ITSC'05)*, Vienna, Austria.
- d'Inverno, M. & Luck, M., 2001. *Understanding Agent Systems*. Springer.
- Daigneau, R., 2011. *Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and Restful Web Services*. Pearson Education.
- Dasheng, W., Ren, L. & Li, J., 2010. *Modeling intelligent transportation systems with multi-agent on SOA*. In 2010 International Conference on Intelligent Computing and Integrated Systems (ICISS), 2010.
- Dia, h. and rose, g. , 1997. *Assessing the performance of artificial neural network incident detection models*, <http://www.uq.edu.au/dia/eur097.pdf> [Accessed 2006 May 11].
- Dramowicz, E, *Three Standard Geocoding Methods*, 2004, Directions Magazines.
http://www.directionsmag.com/printer.php?article_id=670 [Accessed 2009 July 13].
- Ehlert, P. and Rothkrantz, J.M. , 2001. Microscopic traffic simulation with reactive driving agents. *IEEE Intelligent Transportation Systems Conference Proceedings*, Oakland, USA, pp.860-865.
- Eric Bonabeau, 2002. *Agent-based modeling: Methods and techniques for simulating human systems*. In Proc. National Academy of Science., 2002. PNAS.
- Erl, T. et al., 2010. *SOA with .NET and Windows Azure: Realizing Service-orientation With the Microsoft Platform*. Prentice Hall.
- Erl, T. et al., 2012. *SOA with REST*. Prentice Hall.
- Erol, K, Levy, R. and Wentworth, J. *Application of Agent Technology to Traffic Simulation' Intelligent Automation Inc.*
<http://www.tfhrc.gov/advanc/agent.htm> [Accessed 2009 May 7].
- Essen R. V. and Hiestermann V. , 2006. 'X-GDF'-The ISO Model of Geographic Information for ITS'. *ISPRS Workshop on Service and Application of Spatial Data Infrastructure*, XXXVI (4/W6), Hangzhou, China, pp. 14-16.

- Etches, A., Claramunt, C., Bargiela, A., and Kosonen, I., 1998. 'An integrated temporal GIS model for traffic systems'. *GIS Research UK VI National Conference*, March 31-April 2, 1998. University of Edinburgh, UK.
- Exforsys Inc, 2007. *SOA Disadvantages*. [Online] Exforsys Inc Available at: <http://www.exforsys.com/tutorials/soa/soa-disadvantages.html> [Accessed 24 May 2012].
- FHWA, *Simulation tools*, <http://ops.fhwa.dot.gov/trafficanalysistools/index.htm> [Accessed 2010 August 21].
- Garcia-Serrano, A.M. & Vioque, D.T., 2003. FIPA-compliant MAS development for road traffic management with a Knowledge-Based approach: the TRACK-R agents. In *Challenges in Open Agent Systems '03 Workshop*, 2003.
- Giorgini, P., Kolp, M. & Mylopoulos, J., 2001. Multi-Agent Architectures as Organizational Structures. *Autonomous Agents and Multi-Agent Systems*, 13, p.2006.
- Goncalves, J.F., Esteves, E.F., Rossetti, R.J. & Oliveira, E., 2009. *Simulating Communication in a Service-Oriented Architecture for V2V Networks*, 2009. Springer-Verlag.
- Graham, C. et al., 2010., *Market Share: RDBMS Software by Operating System, Worldwide*, [Accesses April 30, 2010]
- Guerra-Hernández, A., Fallah-Seghrouchni, A.E. & Soldano, H., 2004. *Learning in BDI Multi-agent Systems*. Computational Logic in Multi-Agent Systems.
- Hall, D. L. & Llinas, J., 1997. 'An introduction to multisensory data fusion. *Proc. IEEE*, 85(1), pp. 6–23.
- Hameseder, K., Fowler, S. & Peterson, A., 2011. *Performance analysis of ubiquitous web systems for Smartphones*. In 2011 International Symposium on Performance Evaluation of Computer & Telecommunication Systems (SPECTS), 2011. IEEE.
- Hanssen, G.K., Westerheim, H. & Bjørnson, F.O., 2006. Using Rational Unified Process in an SME – A Case Study. [Online] Available at: <http://www.tamps.cinvestav.mx/~ertello/swe/s02-RUPCaseStudy.pdf> [Accessed 16 April 2012].
- Henderson-Sellers, B. & Giorgini, P., 2005. *Agent-Oriented Methodologies*. Idea Group Pub.
- Hernandez, J.Z., Ossowski, S. & Garcaa-Serrano, A., 2002. Multiagent architectures for intelligent traffic management systems. *Transportation Research Part C: Emerging Technologies*, 10, pp.473 - 506.
- Hinchcliffe, D., 2009. *Where Is The Future of SOA Headed? Where The Web Goes*, http://www.ebizq.net/blogs/enterprise/2009/09/where_is_soa_heading_where_the.php [Accessed 13 July 2010].

- Horrocks, I., Patel-Schneider, P.F. & Harmelen, F.v., 2003. *From SHIQ and RDF to OWL: The Making of a Web Ontology Language*. Journal of Web Semantics, 1(1), pp.7-26.
- Huijbers, R. et al., 2004. *Software Project Management: Methodologies & Techniques*. Department of Mathematics & Computer Science, Technische Universiteit Eindhoven.
- Hunter, J. & Lagoze, C., 2001. *Combining RDF and XML schemas to enhance interoperability between metadata application profiles.*, 2001. ACM.
- IBM, 2007. *IBM Rational Unified Process, Improving project performance with proven, adaptable processes*. [Online] Available at: ftp://public.dhe.ibm.com/software/rational/web/datasheets/RUP_DS.pdf [Accessed 16 April 2012].
- Iglesias, C.A., Garijo, M. & Gonzalez, J.C., 1999. *A survey of agent-oriented methodologies*. In Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages. 1999.
- Innocenti B., et al., *Towards seamless real-time in-car and personal navigation service delivery: the HIGHWAY integrated Architecture*, <http://www.ist-highway.org/HIGHWAYIntegratedArchitecture.pdf> [Accessed 2007 Oct 10].
- Jennings, N.R., Sycara, K. & Wooldridge, M., 1998. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1), pp.7-38.
- Juan, T., Pearce, A. & Sterling, L., 2002. *ROADMAP: extending the gaia methodology for complex open systems*. ACM.
- Kaplan, E.D., and Hegarty C.J., 2006. *Understanding GPS: Principles and Applications*, 2nd Edition.
- Katwijk, R.T.v., 2008. *Multi-Agent Look-Ahead Traffic-Adaptive Control*. PhD Thesis. Delft, The Netherlands: TRAIL Research School.
- Kolp, M., Giorgini, P. & Mylopoulos, J., 2006. Multi-Agent Architectures as Organizational Structures. *Autonomous Agents and Multi-Agent Systems*, 13(1), pp.3-25.
- Lake, R, *Introduction to GML Geography Markup Language*, Galdos Systems Inc <http://www.w3.org/Mobile/posdep/GMLIntroduction.html> [Accessed 2009 May 13].
- Larbo M: *Vehicle Navigation: Market Research and Evaluation of Existing Systems*, Master of Science Thesis in Geo-informatics, Royal Institute of Technology, Sweden, April 2000, http://www.geomatics.kth.se/MSc/Exjobb_ML.pdf [Accessed 10/12/2007].
- Lee, S.P., Chan, L.P. & Lee, E.W., 2006. Web Services Implementation Methodology for SOA Application. In *2006 IEEE International Conference on Industrial Informatics*. Singapore, 2006. IEEE.

- Lefevre, K., 2011. *Pitfalls when using PRINCE2®*. [Online] Available at: <http://www.pm4all.be/Content/En/Resources/Blogs/Blogs.aspx?Id=2> [Accessed 20 May 2012].
- Li, L. et al., 2005. 'TVS 05: New Developments and Research Trends for Intelligent Vehicles'. *IEEE Intelligent Systems*, 20 (4), pp. 10–14.
- Li, Y., 2004. 'Journey time estimation and incident detection using GPS equipped probe vehicles' Ph.D. thesis, Southampton University.
- Liu, X. & Fang, Zhiyi, 2008. An Agent-Based Intelligent Transport System. In *Computer Supported Cooperative Work in Design IV*. Springer Berlin / Heidelberg. pp.304 - 315.
- Llinas, J. & Walts, E., 1990. Multi-sensor data fusion. Boston: Artech House.
- Logi, F. & Ritchie, S.G., 2002. A multi-agent architecture for cooperative inter-jurisdictional traffic congestion management. *Transportation Research Part C: Emerging Technologies*, 10, pp.507 - 527.
- Luck, M., 2004. Guest editorial: Challenges for agent-based computing. *Auton. Agents Multi-Agent Systems*, 9(3), pp.199-201.
- Luck, M., McBurney, P., Shehory, O. & Willmott, S., 2005. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink.
- Ma, B., Chen, B., Bai, X. & Huang, J., 2010. *Design of BDI Agent for Adaptive Performance Testing of Web Services*. 10th International Conference on In Quality Software (QSIC).
- Macal, C. M., and M. J. North, 2007. *Agent-based modeling and simulation: desktop ABMS*. In Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come, Washington D.C.
- Madureira, A., Santos, F. & Pereira, I., 2008. Self-managing agents for dynamic scheduling in manufacturing, 2008. ACM.
- Maleshkova, M., 2008. Acquisition and management of semantic web service descriptions. In *The ESWC 2008 PhD Symposium*. Tenerife, SPAIN, 2008.
- Mark Lorenz, 2006. *SOA best practices*. [Online] IBM Available at: <http://www.ibm.com/developerworks/webservices/tutorials/ws-soacert1/section2.html> [Accessed 22 June 2010].
- Matko, D. et al., 2008. The Application of Reference-path Control to Vehicle Platoons., 2008.
- McIlraith, S.A., Son, T.C. & Zeng, H., 2001. *Semantic Web services*. Intelligent Systems, IEEE, , 16, pp.46 - 53.

- Milojicic, D., 1999. 'Trend Wars - Mobile agent applications Concurrency' *IEEE Intelligent Systems*. 7(3), pp. 80-90.
- Mirchandani, P. & Wang, F.-Y., 2005. RHODES to intelligent transportation systems. *Intelligent Systems, IEEE*, 20, pp.10 - 15.
- MSDN, 2007. *SOA in the Real World*. [Online] Microsoft Available at: <http://msdn.microsoft.com/en-us/library/bb833022.aspx> [Accessed 20 June 2010].
- Niver, E. et al., 2000. Evaluation of TRANSCOM's system for managing incidents and traffic (TRANSMIT). *IEEE Transactions on Intelligent Transportation Systems*, 1, pp. 15-31.
- Odell, J., 1999. *Objects and Agents: Is There Room for Both?* Complex Adaptive Systems, pp.44-45.
- Odell, J., Van Dyke Parunak, H. & Bauer, B., 2001. *Representing Agent Interaction Protocols in UML*. pp.201-18.
- Oracle, 2010. *SOA Reference Architecture*. SOA Training Slides. Rugby: Oracle.
- Ozbay, & Kachroo, P., 1999. *Incident Management in Intelligent Transportation Systems*. Artech House.
- Ozdag, M., 2007. *Multi-Agent Software Design on Windows*. [Online] Available at: http://www.codeproject.com/KB/architecture/Multi-Agent_Software.aspx?display=Print [Accessed 13 May 2010].
- Parunak, H.V.D., 1999. *Industrial and Practical Applications of DAI*. Multiagent System (AGENTS'01), MIT Press.
- Pitt, J., Mamdani, A. & Charlton, P., 2001. *The open agent society and its enemies: a position statement and research programme*. Telematics and Informatics, 18, pp.67 - 87.
- Ricci, A., Buda, C. & Zaghini, N., 2007. *An Agent-Oriented Programming Model for SOA & Web Services*. In 5th IEEE International Conference on Industrial Informatics. IEEE. pp.1059 - 1064.
- Richardson, L. & Ruby, S., 2007. *RESTful Web Services*. O'Reilly.
- Roozmond, D.A., 2001. Using intelligent agents for pro-active, real-time urban intersection control. *European Journal of Operational Research*, 131, pp.293-301.
- Russell, S.J. & Norvig, P., 2010. *Artificial Intelligence: A Modern Approach*. Prentice Hall.
- Sabine, M. & Karlheinz, K., n.d. *Applying system development methods in practice*. [Online] Available at: <http://www.rupsoa.com/wp-content/uploads/Library/RUP/ABRUP.pdf> [Accessed 14 April 2012].

- Sabo, M., 2006. *Building Web Service Ontologies*. PhD Thesis. SIKS, the Dutch Graduate School for Information and Knowledge System.
- Sanchez Passos, L. & Rossetti, R., 2010. *Traffic light control using reactive agents*. In 2010 5th Iberian Conference on Information Systems and Technologies (CISTI), 2010.
- Schwab, A., 2007. *Synchronising Via SOA - Using a Service-Oriented Architecture to Synchronise Mobile Agents*. VDM Verlag.
- SCOOT-UTC, n.d. *INGRID Overview*. [Online] Available at: <http://www.scoot-utc.com/INGRID.php> [Accessed 13 Dec 2009].
- Shehory, O. & Sturm, A., 2001. Evaluation of modeling techniques for agent-based systems., 2001. ACM.
- Siemens Mobility, 2009. *SCOOT USER GUIDE*. POOLE: SIEMENS PLC.
- Siemens, 1999. *Results from SCOOT's Commercial Systems*. Published Results. Siemens.
- Simonin, O. & Gechter, F., 2005. *An Environment-Based Methodology to Design Reactive Multi-agent Systems for Problem Solving*. E4MAS.
- Srinivasan, D. & Choy, M., 2006. Cooperative multi-agent system for coordinated traffic signal control. *IEE Proceedings - Intelligent Transport Systems*, 153, pp.41-50.
- Strahle, M. et al., 2007. Towards a Service-Oriented Architecture for Interconnecting Medical Devices and Applications. In *HCMDSS-MDPnP Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability*. Boston, 2007.
- Tagni, G.E. & Jovanovic, D., 2006. *Comparison of Multi-Agent Systems JACK vs 3APL*. [Online] Available at: http://www.tagni.com.ar/docs/jack_3apl.pdf [Accessed Aug 2010].
- Tang, S., Wang, F.-Y. & Miao, Q., 2006. ITSC 05: Current Issues and Research Trends. *IEEE Intelligent Systems*, 21(2), pp. 96–102.
- Tapia, D.I., Rodríguez, S., Bajo, J. & Corchado, J.M., 2008. FUSION@, A SOA-Based Multi-agent Architecture. In *In Proceedings of DCAI'2008.*, 2008.
- Trivedi, M., 2001. *ITTS-Research@UCSD, Intelligent Transportation and Telematics Layer, Joint (UCI+UCSD) in Technical Group Meeting*. [Online] University of California at San Diego.
- Trullàs-Ledesma, J. & Ribas-Xirgo, L., 2009. *From SOA to MAS: migrating an architecture for mobile devices applications*. X workshop de agentes físicos, cáceres.

- Vecchiola, C., Gozzi, A., Coccoli, M. & Boccalatte, A., 2008. *An Agent Oriented Programming Language Targeting the Microsoft Common Language Runtime*. Genova, Italy: University of Genova.
- Vidal, J.M., 2010. *Fundamentals of Multiagent Systems with NetLogo Examples*.
- Wang, D., Ren, L. & Li, J., 2010. Modeling intelligent transportation systems with multi-agent on SOA. *2010 International Conference on Intelligent Computing and Integrated Systems (ICISS)*, pp. 717-720.
- Wang, D., Ren, L. & Li, J., 2010. *Modeling intelligent transportation systems with multi-agent on SOA*. In 2010 International Conference on Intelligent Computing and Integrated Systems (ICISS), 2010.
- Wang, F.-Y., 2005. Agent-based control for networked traffic management systems. *IEEE Intelligent Systems*, 20(5), pp.92- 96, Sept.-Oct.
- Wang, F.-Y., 2006. Driving into the Future with ITS. *IEEE Intelligent Systems*, 21(3), pp.94-95.
- Wang, F.Y., 2008. Toward a revolution in transportation operations: AI. *IEEE Intell. Syst.*, 26(6), p.8–13.
- Weiss, G., 2000. *Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press.
- Wells, D., 2009. *Extreme Programming: A gentle introduction*. [Online] Available at: <http://www.extremeprogramming.org/> [Accessed 13 May 2012].
- Wooldridge, M., 2009. *An Introduction to MultiAgent Systems*. John Wiley & Sons.
- Wooldridge, M., Jennings, N.R. & Kinny, D., 2000. *The Gaia Methodology for Agent-Oriented Analysis and Design*. *Autonomous Agents and Multi-Agent Systems*, 3, pp.285-312.
- Wu, Z. et al., 2005. DataGrid II: A semantic grid platform for ITS. *Intelligent Systems, IEEE*, , 20, pp.12 - 15.

End of thesis

Appendices

- Appendix **A**: Commercial implementations & contributions
- Appendix **B**: Other research implementations & contributions
- Appendix **C**: Intelligent Transportation Systems (ITS) Review
- Appendix **D**: Agent Oriented Technologies Review
- Appendix **E**: Ontology languages overview
- Appendix **F**: Review of the Tools & Technologies relevant to this research
- Appendix **G**: Wireless Communication Technologies Review
- Appendix **H**: ITS@CU Platform Components design details
- Appendix **I**: Mobile Application Development Framework (MADF) Description
- Appendix **J**: CD Contents and project Source Code
- Appendix **K**: NetLogo simulation study for traffic evaluation
- Appendix **L**: SCOOT data and traffic area study report
- Appendix **M**: Schema XSD – Agent Message Structure
- Appendix **N**: Schema XSD – Semantic Content Structure
- Appendix **O**: ITS@CU Ontology Structure
- Appendix **P**: ITS@CU Applications Screenshots

Appendix A: Commercial implementations & contributions

As mentioned in *chapter 1*, this research started as part of a KTP funded project in association with T@lecom who are one of the leading providers of asset tracking, transportation and logistics related mobile software solutions in the UK. The research project had a commercially-oriented approach targeted towards T@lecom's R&D efforts to implement a new platform for the development of a new generation of mobile solutions and ITS-based systems. In addition to the core research objectives, the R&D efforts also led to the development of various commercial applications and Proof of Concepts (PoC).

Some of the relevant commercial implementations which resulted from this research are described below:

1. Intelligent Vehicle Tracking System

It is a GPS based commercial asset tracking system developed from elements of the research project. The system provides a complete tracking solution for different type of assets such as vehicles and mobile devices. It comprises of different components i.e. a mobile application; data collection and notification web services; central database; and interactive map based monitoring and management application.

Background

In the early stage of this research, the author developed a web based mapping and simulation application based on Microsoft Virtual Earth intended for the ITS@CU platform's simulation and also to test an experimental Incident Detection System using real-time GPS data (outlined in appendix B). Due to the application's GPS based tracking features, T@lecom's CTO (who was also an industrial KTP advisor) at that time asked the author to extend the simulation system for commercial vehicle tracking purposes. T@lecom at the time was using a proprietary 3rd party vehicle tracking solution for its customers which was costly and also lacked features such as web based monitoring and up to date satellite maps. After 4 months of development effort a pre-release version was developed by the author with real-time vehicle tracking, geo-fencing and history trails features. This version was assessed rigorously by T@lecom's software development team, and also by their business analysts for its business case viability for commercial implementation. The system was further enhanced and deployed on a trial basis for tracking a customer's fleet of 20 large vehicles delivering fresh food supplies. The system since then has been evolved and used by various T@lecom's customers.

It is now integrated with T@lecom's WD¹ Live solution including the Patient Transport System for non-emergency ambulances.

¹ **WD** stands for Wireless Delivered and it is a registered trademark of T@lecom Limited. Most of the mobile solutions and products by T@lecom use WD as part of their branding.

Note: This system is now called WD Live Tracking and the IPR is owned by T@lecom. It is not part of the “Wire3” which is another tracking product from T@lecom.

Technical Description

The system comprises of the following components:

- A. Mobile Tracker Application
- B. Data & Notification Service
- C. Central Database
- D. Asset Monitoring & Management web Application

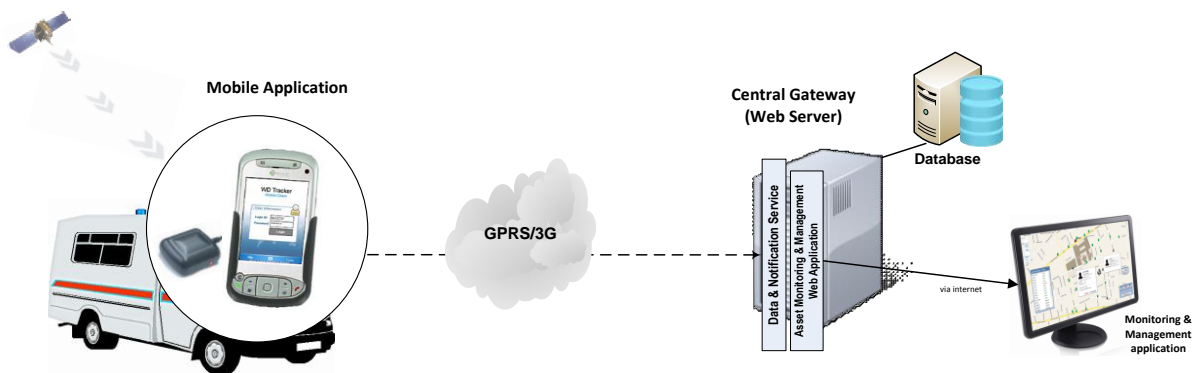


Figure 1.1: Tracking System Overview

A. Mobile Tracker Application

It is a PDA/mobile application (called WD Tracker Mobile) which can be installed on a range of Windows Mobile 5/6.1 PDA devices. The application is used by vehicle drivers or field users during their work shifts on their work PDA devices. The driver/user logs in to the application with their User ID or vehicle registration on the start of their shift. Each user belongs to a group or company, and the application on login determines the features set and access level based on the user's role.

The application uses real-time GPS data obtained from a GPS receiver either built-in within PDA or externally fitted. During operation, the application displays useful GPS data such as current position, Speed, Direction and connectivity status in real-time. It also provides feature to control several Navigation applications such as TomTom

5/6 and Co-Pilot 7. Using this feature the driver can start navigating right from the mobile application to a pre-set destination (send by the control room system).

In the background the application collates GPS signals (National Marine Electronics Association NMEA) obtained by the receiver and converts it to World Geodetic System - WGS-84 format, which is the more standardised format understood by most GPS or location aware applications. The data is then periodically sent over GPRS/3G to the Data & Notification Web Service (hosted on the gateway server). The application has various connection management features to support the occasionally connected nature of the application and deals with out of mobile network coverage and GPRS issues. In addition, the application uses Microsoft Message Queuing (MSMQ) software which queues all the data in device memory before sending it to the gateway server. In the event of connection loss, the data is queued locally and as soon as device 3G/GPRS connection is re-established the data is sent to the server in an orderly fashion.

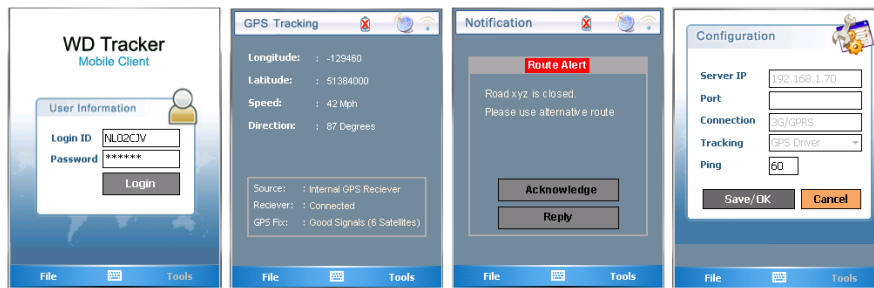


Figure 1.2: Mobile Application login, GPS status, Alert/Notification messages and Configuration Screens

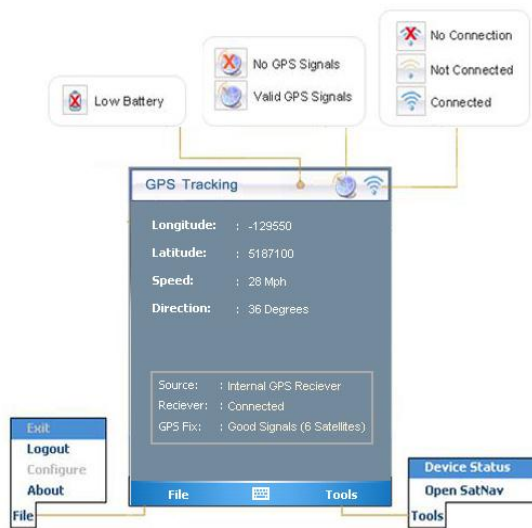


Figure 1.3: Mobile Application Main Screen

One of the key features of this mobile application is the capability to receive messages and notifications send by the control room staff. The drivers/users must acknowledge any incoming messages, which automatically send a confirmation back to the control room server that the message was read by the user. It also allows the user to reply to such messages (in free form text) enabling two-way messaging. This feature allows the Control room staff to communicate with the field drivers by sending alerts/notifications to either an individual or broadcast to a group of

drivers/vehicles. This feature reduces the need to call the driver's phone hence reduce the operating cost dramatically.

The application is fully configurable and the interval of GPS data update, type of receiver, type of vehicle and other log management features can be configured by an authorised user (admin role).

The application was developed in C# and .Net Compact Framework 2.0. It also uses some core libraries as part of Windows Mobile 5 API written in C++. It also store data locally using Microsoft SQL Server Compact Edition 3.5. The database and the data transaction over the air (GPRS/3G) is fully encrypted using AES-128 level encryption for high security. The web service based communication is compressed using GZIP compression by using SOAP extension.

B. Data & Notification Service

This is a Web Service hosted on the T@lecom's Central Gateway web server. It provides a communication interface for all the incoming and outgoing transactions to and from PDA devices to the Server. *It performs the following functions:*

- Receives the GPS data from all the mobile applications
- Stores the GPS data in relevant databases on the Server
- Sends Alerts and Notification messages to mobile application on PDA device(s)
- Maintain sessions and stores device connectivity & errors logs
- Provides an integration capability for external systems

This is SOAP based web service application hosted on web server's Internet Information Services (IIS). It is developed using ASP.NET 2.0 and C#. The SOAP classes are modified (SOAP extension) allowing GZIP based compression and AES-128 encryption.

C. Database

This is the backend database server (running SQL Server 2008 R2 Enterprise Edition) with databases for storing all the information related to assets (vehicle, users and PDAs), GPS data, messages/alerts, groups and companies, and other information required for the entire system. Following is the database schema diagram for the tracking system.

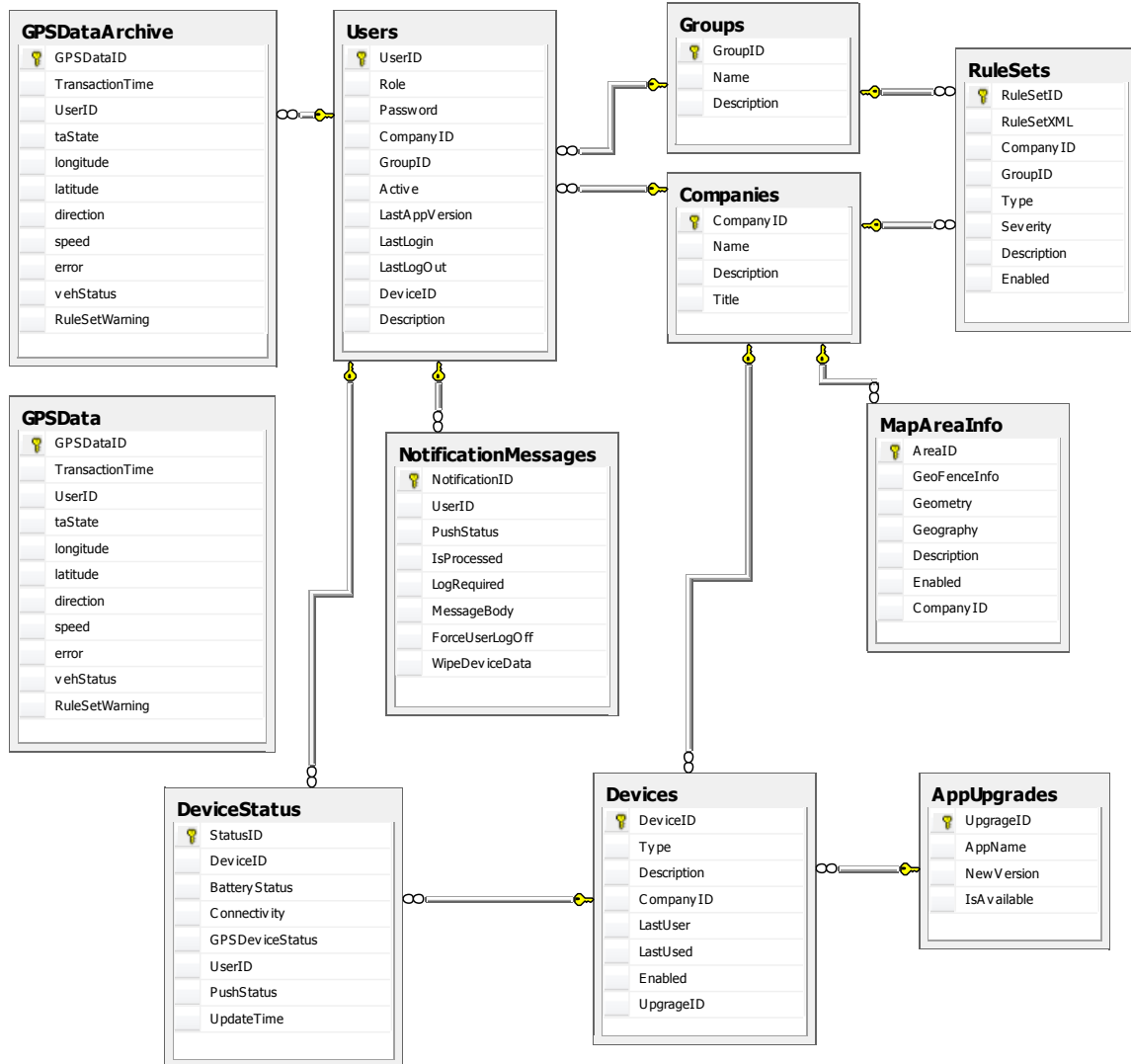


Figure 1.4: Database Model

The database also uses Spatial Data (also known as geospatial data or geographic information) which is a new GIS feature of SQL Server to identify the geographic location of features and boundaries on Earth, such as natural or constructed features (e.g. roads, buildings, landmarks, rivers etc.). Spatial data is stored as coordinates and compatible with most mapping solutions such as Bing, MapPoint and Google Maps.

All the SQL queries are stored as stored procedures allowing high level of flexibility and optimisation. The database supports replication between PDA devices and also between sites for resiliency purposes.

D. Asset Monitoring & Management web application

This is the main application with interactive mapping interface for the back-office staffs (operators and administrators) to track/monitor, communicate and manage their field assets (vehicles, field force/users, devices etc.). It is a web-based application hosted on a central gateway web server and customer access it over internet using their web browser such as Internet Explorer 7/8, Firefox 3.x or Chrome. The application is highly secure using HTTPS, password protected and accessible from only allowed IP address ranges.

The application can be used by multiple users/customers at the same time however each user only views their own assets (belong to their group or company). For example an “operator” user of company XYZ can only view the vehicles of company XYZ, and an “admin” user of company XYZ can only manage the vehicles, PDAs, users and other admin features related to company XYZ.

The application starts with a login screen and based on the user’s affiliation and role the application displays relevant features and assets allowed for that login. The application has a map based interactive UI and touch screen friendly (see *figure 1.5*). After successful login, the application shows the current location and status of the vehicles/assets:

- **Active:** Vehicle is currently on a shift or the mobile application is in use.
- **Inactive:** This indicates vehicle is not currently on a shift.
- **Warning:** This status highlights various other underlying statuses such as user outside of designated area/geo-fence; over speeding; Notification/Alert not acknowledged etc. These are configurable rule-sets defined by the administrator of the system.

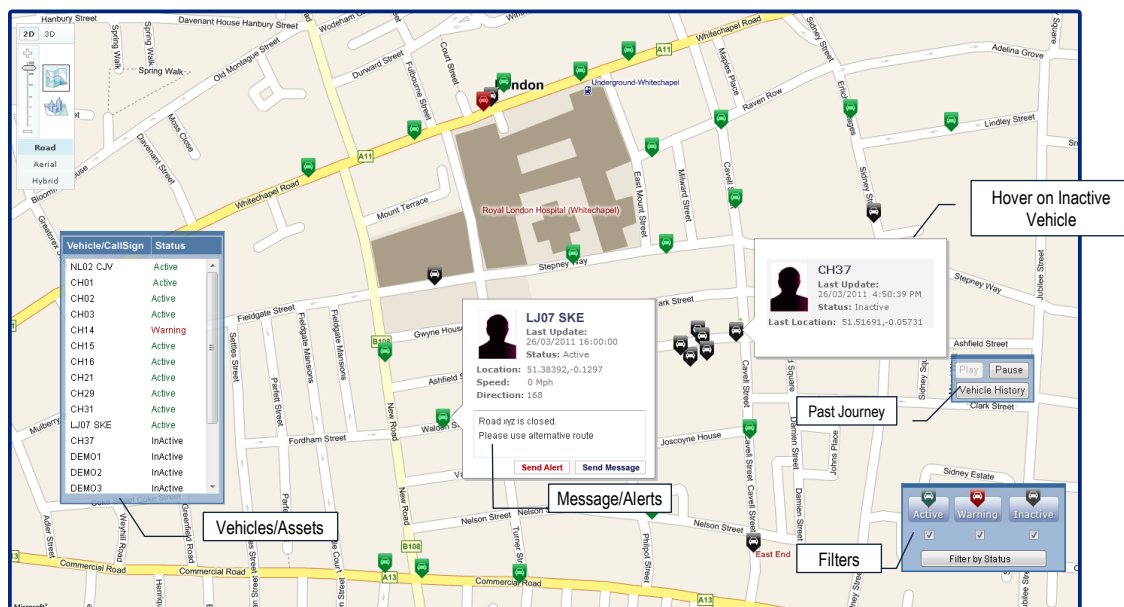


Figure 1.5: Asset Monitoring View

The user can manipulate map location, zoom level, views (map, hybrid, satellite, bird's eye) and filter/display assets by current status. Hovering mouse pointer on any asset icon displays a pop up window with vehicles location description, and if required a message or alert can be send to that user's mobile application.

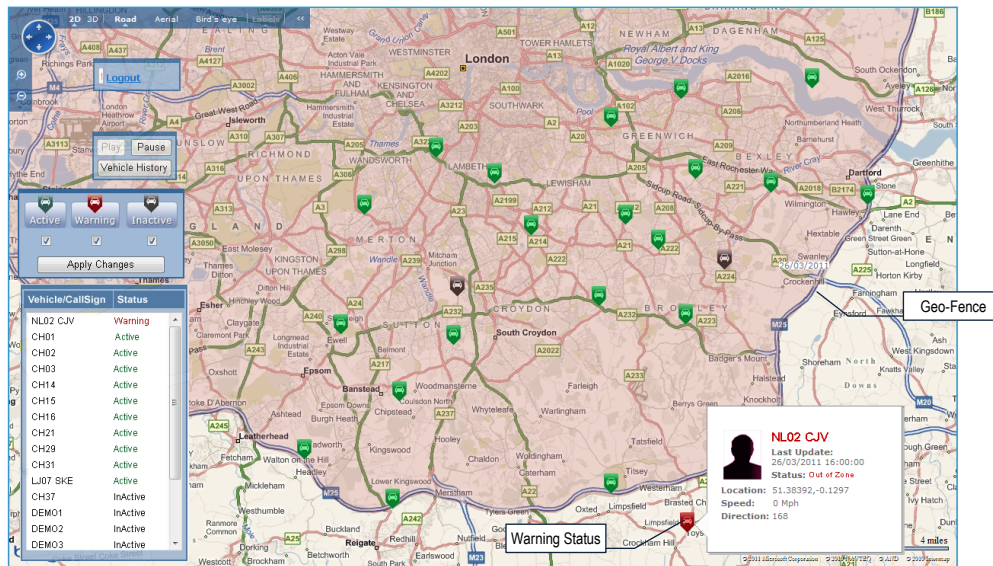


Figure 1.6: Geo-Fence based Asset Monitoring View

One of the other main features of the system is to set Geo-fences (customised map boundaries) and each user can be assigned to a geo-fence (see figure 1.6). So if a vehicle bound to a particular geo-fence goes outside that geo-fence boundary area then the vehicle's status changes to "Warning" status and an alert message is sent to the driver's mobile application (if configured by the administrator).

The system also allows viewing vehicle's past journey (historical tracking information). The user can select a vehicle and enter number of last hours to display the trail of past journey. The user can hover the mouse pointer on each point of the trail to view GPS information obtained during that point (see figure 1.7).

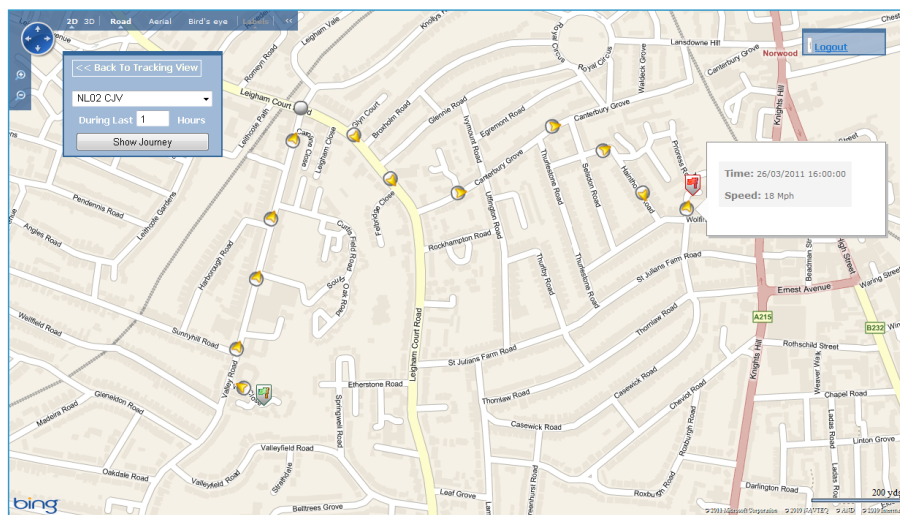


Figure 1.7: Vehicle past journey trail view

The application also has an administration and management view which is only accessible to customers' admin account users, allowing:

- Asset (user, vehicles and PDAs) management
- Geo-fence allocation and manipulation
- Rule set creation for Alerts and Notification
- Users Grouping
- GPS and Message data analysis
- Custom Reports by driver, vehicle, journeys during any period

This application was developed in C# and based on ASP.NET 2.0 framework. It uses Microsoft Virtual Earth 4.0 API (now called Bing Maps). The application also uses AJAX.NET which allows real-time map status without refreshing the web page. This technology makes the UI very smooth and the vehicle location and their status are always kept up to date in real-time without any interruption.

The application interacts with Central SQL Server 2008 Database and also utilises the Data and Notification Web Service for sending Alerts and Notification messages to the PDA application. The application is designed on Model View Presenter (MVP) design pattern providing a robust and flexible design approach.

Commercial benefits

The WD Tracking System is a value added solution as part of T@lecom's WD Live Platform. The system has been used by various customers ranging from logistics to Non-Emergency Ambulances. It has also reduced T@lecom's dependency on 3rd party supplier and proved to be profitable.

Novelty

The system delivers all the features of a modern tracking solution but it also combines a two-way communication and notification features right within the Map based UI. This at the time of implementation was quite a unique feature differentiating it from other such systems, and from a commercial point of view it has been the main marketing feature and highlight of the product.

From technological perspective, the system used industry leading technologies such as Microsoft VE/Bing Maps API, AJAX.Net, Windows Mobile 5/6, SQL Server 2008 Spatial Data and other .NET related technologies in a well-integrated manner. The design of the overall system follows core SOA principles and individual components uses industry standard design approach MVP.

An additional feature of the system is the ability to analyse the driver's driving behaviour such as average speed, acceleration and deceleration patterns. This feature is available in the reporting component of the management application. It uses bespoke algorithms to analyse the historical GPS data and based on that formulate the speeding, acceleration and deceleration patterns of the driver. This feature helps the operation staff to analyse the drivers' behaviour, and proactively avoid vehicle misuse and also reduce the fuel and vehicle maintenance costs. The latter is particularly topical and could be used to reduce Co2 by informing drivers of their past performance in view to improve their driving behaviour.

Challenges

The development of the tracking system had various challenges such as short time period, lack of development resources and technical challenges to name few. Although the author had a solid software development background and industrial experience however the overall duration (8 months) and the scale of the system involving various technologies and wide range of skill set (programing, databases, network, design etc.) was a difficult undertaking. To overcome these challenges, the project was split into smaller components and where ever required the author acquired assistance from the wider development team.

Relevance to the research

Although this system was not the core part of the research, however it greatly helped the research in following ways:

- Various components of the tracking system such as map services, GPS tracking database, mobile application libraries (DLLs) were reused in ITS@CU implementation.
- The map based agent monitoring application is based on the same technologies and design principles
- The mobile application uses the various features/components of the Mobile Application Development Framework (MADF).
- Acquired valuable knowledge/experience in various tools and technologies which were also useful for the development of ITS@CU platform.

2. Patient Transport System

Patient Transport System (PTS) is a mobile software solution by T@lecom for non-emergency ambulance sector. It includes a mobile application and gateway services integrated with NHS control centre system via T@lecom's Mobile Gateway System. The mobile application is used by ambulance crews to transport patients to and from hospitals. It provides work/task lists, patient information, notification features, integrated Navigation (TomTom 6), GPS tracking, and real-time exchange of status information between the ambulance crew and their control centre.

Background

Mobile Application Development Framework (MADF) was one of the objectives developed as part of the research (discussed in *chapter 7, section 7.4*). As the new framework was very flexible and based on robust latest technologies, T@lecom assigned the author to evaluate and develop a Proof of Concept (PoC) of their PTS mobile application using this new framework. The existing PTS application was based on older Pocket PC 2003 and .NET compact framework 1.0 and it provides an ideal development framework to upgrade the PTS mobile application.

The PoC application was developed in approximately 3-4 months which was evaluated internally by T@lecom's test and software development team. It was further enhanced by the author (3 more months' effort) and then handed over to T@lecom's software development team for final development.

Remark: The PTS system is quite a large system and includes a mobile application, set of gateway services, databases and back-end system integrators/connector services. The contribution of this research is mainly focused on the PTS mobile application and therefore rest of this section only outlines the early contributions made by the author which are relevant to the research efforts.

Technical Description

PTS system comprises of the following components:

- A. PTS Mobile Application
- B. Web Services
- C. Database server and device replication

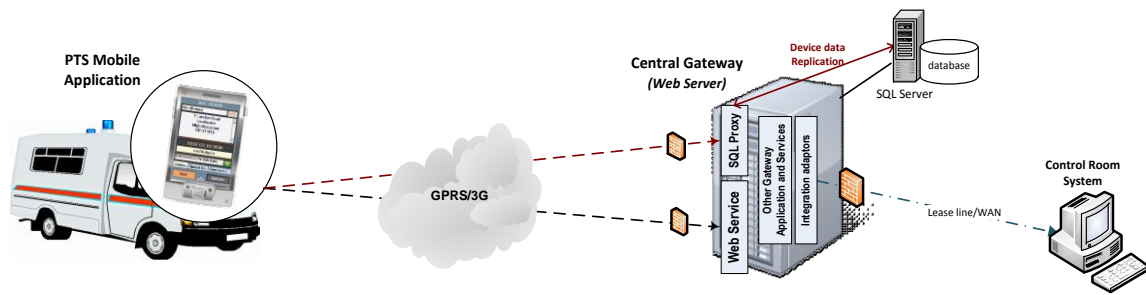


Figure 1.8: PTS System Overview

A. PTS Mobile Application

It is a mobile application (*also called WD PTS Mobile*) which can be installed on a range of Windows Mobile 5/6.1 based PDA devices. The application is used by ambulance crews during work shifts using their work PDAs. A crew member logs in to the application with their credentials (see *figure 1.9*). The application after successful login communicates (over GPRS/3G) with the web services hosted on T@lecom's gateway server in order to download a list of work tasks assigned to the crew (see *figure 1.10*).



Figure 1.9: Login and job details screen

Note: The tasks generally are a list of patients which require pick-up or drop-off from hospitals or homes. These tasks (or jobs lists) are dynamically updated and assigned by the control room system based on crew's location and availability status. The work tasks are stored on the gateway server's database and kept up to date by an integration adapter application developed by T@lecom.

PTS mobile application displays the tasks/jobs list in an order specified by the control room system (see *figure 1.10*). Using the application, the crew selects individual tasks to perform their job routine and update task status (e.g. arrived at patient's pick-up location, patient on-board, patient not available, drop off, lunch break, emergency etc.). The status updates are transmitted to control room in real-time which helps control room staff monitor the crews work activities and tasks progress in real-time. The application provides various other features such as on-board

patient information, in-app calling and SMS, emergency notification etc. It also provides navigation feature (using TomTom 5/6) allowing the driver to start navigating to patient's pickup or drop off location (send by the control room system as part of job list). The application also allows messaging and notifications to and from control room system. This feature allows the Control room staff to communicate with individuals or broadcast to a group of ambulances. This feature reduces the need to call the driver's phone hence reduce the operating cost dramatically.

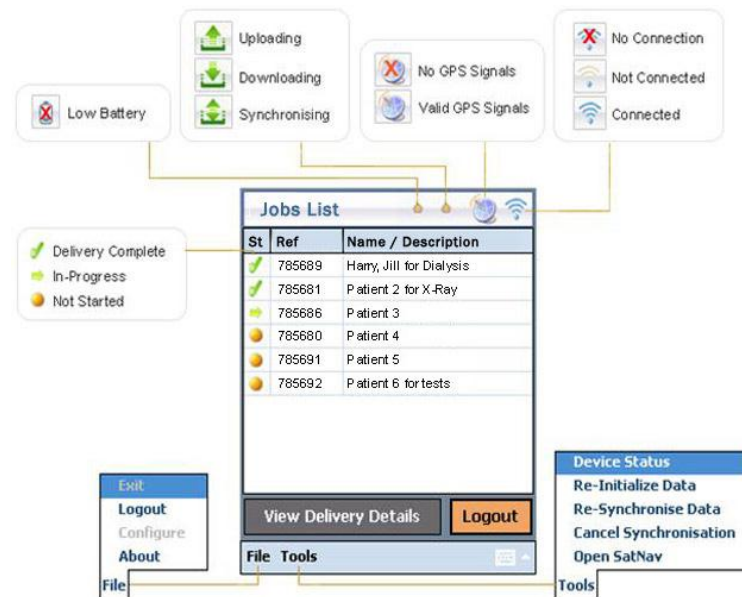


Figure 1.10: Tasks/Jobs list screen with test data

The mobile application uses two methods to communicate with the gateway server 1) Web services and 2) Merge Replication. Web Services interface provides general features such as user authentication, messaging, status updates, GPS data, application updates, job/tasks updates check etc. Merge replication synchronises the device database with server database so if any data changes either on server or device the replication mechanism keeps both datasets up to date.

The mobile application has various connection management features to deal with mobile network coverage and GPRS connectivity issues. In addition, the application uses Microsoft Message Queuing (MSMQ) software which queues all the data in device memory before sending it to the gateway server. In the event of connection loss, the data is queued locally and as soon as device the 3G/GPRS connection is re-established the data is sent to the server in an orderly fashion. The mobile application was designed to be configurable and the interval of GPS data update, application update and log management features can be configured by an authorised user (admin role).

The design of the overall system follows core SOA principles and individual components uses industry standard design approach Model View Presenter (MVP).

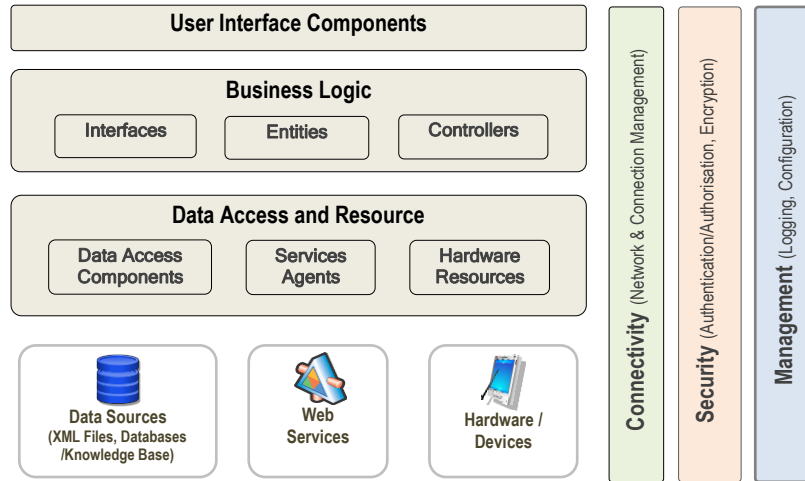


Figure 1.11: PTS Mobile application Design (based on MVP)

The application was developed in C# and .NET Compact Framework 2.0. It also uses various core C++ libraries (DLLs) as part of Microsoft Windows Mobile 6.5 API. The application stores data locally using Microsoft SQL Server Compact Edition 3.5. The local database and the data transactions over the air (GPRS/3G) are encrypted using AES-128 level encryption for high security. The web service transactions are compressed using GZIP compression by using SOAP extension technique.

B. PTS Web Service

This is the main web service hosted on T@lcom's Central Gateway web server. It provides a communication interface for all the incoming and outgoing transactions to and from PDA devices to the gateway server. *It performs the following functions:*

- Validates login credentials of the mobile application users
- Sends Alerts and Notification messages to mobile application on PDA device(s)
- Notify mobile applications when Tasks/Jobs lists are updated on server which triggers Merge Replication process on the relevant device
- Receives periodic status updates and GPS data from all the mobile applications, and stores the data in relevant database on SQL Server
- Stores device connectivity & errors logs

The web service is SOAP based and hosted as a web application on Internet Information Services (IIS) on the web server. It is developed using ASP.NET 2.0 and C#. The SOAP classes are modified (SOAP extension technique) allowing GZIP based compression and AES-128 encryption. The services are designed on SOA principles and

layered approach (presentation, business, access, integration etc.), service brokers, orchestration, service composition and allocation.

C. Database Server and Merge Replication service

PTS solution has a staging database for storing partial data containing the current jobs/tasks allocated to ambulance crews (for a limited period). The staging data is pulled from Control Room System by a separate integration adapter service (already developed by T@lecom). The data is downloaded by the mobile application using Merge Replication (a synchronisation method by Microsoft SQL Server 2008). The gateway web server hosts a SQL Server synchronisation proxy which allows mobile application's to synchronise SQL Server 3.5 compact edition on the device with SQL Server 2008 on the server over HTTPS. Merge Replication is a highly secure and robust mechanism to update the tasks/jobs list and can handle large amount of data seamlessly.

There are other components supporting the PTS solution such as "Gateway services" and "Control Room integration adaptor service" which are not discussed here as they were developed separately by T@lecom and not relevant to the research.

Commercial benefits & Novelty

The Mobile Application Development Framework developed as part of the KTP research provided a flexible development approach for upgrading T@lecom's existing PTS mobile application. The author was predominantly involved in the development of pre-release version which eventually led to the final application now commercially being used by various ambulances services.

The new version provides better application UI performance and also communication reliability especially the data download over slow and unstable GPRS mobile networks. In addition to the existing features, the new version also provides a two-way communication and notification features.

It also uses standard SOAP based web services (compare to proprietary connection method used in the previous version) which allows the application to take advantage of T@lecom's SOA platform and ability to scale the system rapidly. The Merge Replication (on Windows Mobile) at the time of development was a new technology allowing highly robust synchronisation of large amount of data securely over wireless (GPRS/3G) connection which exceeded the expectations of the performance requirements.

From a technological perspective, the system used industry leading technologies at that time such as Microsoft Compact Framework 2.0, ASP.NET, Windows Mobile 5/6.5 APIs, SQL Server 2008, SSQ Server compact edition 3.5, Merge Replication, TomTom in-app integration API, SOAP extension for compression and security other .NET related technologies in a well-integrated manner.

Challenges

The development of PTS Proof of Concept application was a challenging task. The main requirement was to make significant performance improvements (UI and data transmission) without compromising any of the existing features. Reliable connectivity in a poor mobile network coverage area (especially GPRS coverage) was a major issue which becomes worse in buildings, moving vehicles and remote areas. Ambulance crew deals with all such situations during their work. The challenge was three fold i.e. reduce data size, queue data locally and automatically re-establish connection all seamlessly without affecting application.

The author tested various technologies/methods to reduce data sizes and adopted GZIP compression by customising web service to reduce the SOAP overheads. In addition, Microsoft Message Queuing (MSMQ) software was used to queue all the data in device memory before sending it to the gateway server. In the event of connection loss, the data is queued locally and as soon as device 3G/GPRS connection is re-established the data is sent to the server in an orderly fashion. Security was also an issue with the web service. A common approach was the use of Microsoft's Web Service Enhancement (WSE 3.0) however it was designed for high bandwidth connection not ideal for mobile environment and therefore had significant performance implications. After various tests the author implemented customised classes for utilising AES-128 encryption and modified web service transaction layer using SOAP extension technique.

Relevance to the research

PTS application was developed based on MADF which was a core part of the research objectives. The application provided commercial benefits and helped the author learn new tools/technologies which were also used in the ITS@CU platform development. Some of the mobile application libraries (DLLs) developed for this application were reused in ITS@CU simulation applications such as MSMQ, SQL server compact edition Merge Replication and GPRS connection management classes.

3. Mobile Gateway System

These are a set of web services for unified mobile applications communication layer and a web-based management application hosted on T@lecom's Gateway System for managing and monitoring mobile devices. The gateway services were initially developed as part of the ITS@CU platform for the communication between mobile devices and the gateway server. T@lecom assigned the author to extend this initial gateway services layer so it could be used for all other mobile applications (built on the new Mobile Application Development Framework) and also to integrate with T@lecom's new SOA platform. Additionally, a web based utility was also required to view and manage the current field devices status using the Gateway Services. The author spent estimated 3 months on the gateway services and management application, and since then it has been further improved by T@lecom development team due to various customer requirements and change requests.

Technical Description

The system comprises of the following components:

- A. Gateway Services
- B. Gateway Management Application

A. Gateway Services

A set of web services (hosted on T@lecom's Gateway web server) providing a communication interface for all the incoming and outgoing transactions to and from PDA devices to the gateway server. *It performs the following functions:*

- Provides customer business processes (customised service workflows)
- External services (Address search, direction/location info etc.)
- Validates login credentials of the mobile application users
- Sends Alerts and Notification messages to mobile application on PDA device(s)
- Notify mobile application if any updates available on the server
- Receives periodic status updates and GPS data from all the mobile applications, and stores the data in relevant databases on the server
- Maintain device sessions
- Stores device connectivity & errors logs

It combines SOAP web services and Windows Communication Foundation (WCF) services into composite workflows. The services layer was designed on SOA principles and layered approach (presentation, business, access, integration etc.), service brokers, orchestration, service composition and allocation.

These services are hosted on Internet Information Services (IIS) on the gateway server. It was developed using ASP.NET 2.0, WCF and C#.

B. Gateway Management Application

It is a web based application primarily developed for T@lecom's administrators/operators to monitor the gateway communication activities, service usage and manage mobile applications/devices. It provided a flexible way to manage the gateway remotely by multiple users as compared to the existing management software which was only accessible on the server where gateway services were hosted.

It performs the following functions:

- Monitor real time status of the users/vehicles and mobile devices
- Asset management (Users and PDAs devices)
- Send messages and Alerts/Notifications to individual users or broadcast to group
- Generate custom reports (usage/activities by driver/user, vehicle during a specified period)
- Download and view mobile device error logs for diagnosis/trouble shooting
- Over the Air upgrades for mobile applications
- Other system admin tasks (data clean up, system services analysis and configuration etc.)

WD Gateway Management

administrator | Logout

User/Device Status | Data Push | Upgrades | Logs | Users | Tracking | System Monitor | Admin

Receive Time	User	Device Time	GPS Status	Battery	Device ID/IMEI
05/05/2011 15:05:38	NL02CJV	07:03:29	GPS/ON/Signal OK	0	
05/05/2011 15:05:20	NL02CJV	07:04:11	GPS/ON/Signal OK	0	
05/05/2011 15:05:11	NL02CJV	07:04:16	GPS/ON/Signal OK	0	
05/05/2011 15:05:01	NL02CJV	07:02:59	GPS/ON/Signal OK	0	
05/05/2011 15:04:51	NL02CJV	07:02:54	GPS/ON/Signal OK	0	
05/05/2011 15:04:41	NL02CJV	07:02:49	GPS/ON/Signal OK	0	
05/05/2011 15:04:31	NL02CJV	07:02:44	GPS/ON/Signal OK	0	
05/05/2011 15:03:15	NL02CJV	07:02:39	GPS/ON/Signal OK	0	
05/05/2011 15:03:10	NL02CJV	07:02:34	GPS/ON/Signal OK	0	
05/05/2011 15:03:05	NL02CJV	07:02:29	GPS/ON/Signal OK	0	
05/05/2011 15:02:59	NL02CJV	07:02:24	GPS/ON/Signal OK	0	
05/05/2011 15:02:55	NL02CJV	07:02:19	GPS/ON/Signal OK	0	
05/05/2011 15:02:49	NL02CJV	07:02:14	GPS/ON/Signal OK	0	
05/05/2011 15:02:45	NL02CJV	07:02:10	GPS/ON/Signal OK	0	
05/05/2011 15:02:42	NL02CJV	07:02:07	GPS/ON/Signal OK	0	
05/05/2011 15:02:34	NL02CJV	07:02:03	GPS/ON/Signal OK	0	
05/05/2011 15:02:29	NL02CJV	07:01:53	GPS/ON/Signal OK	0	
05/05/2011 15:02:24	NL02CJV	07:01:48	GPS/ON/Signal OK	0	
05/05/2011 15:02:18	NL02CJV	07:01:43	GPS/ON/No signal	0	
05/05/2011 15:02:14	NL02CJV	07:01:38	GPS/ON/Signal OK	0	
05/05/2011 15:02:08	NL02CJV	07:01:33	GPS/ON/Signal OK	0	

12345678910...

Device current status information, status grids and charts/graphs, Customised Reports

Vehicle/Device tracking (opens tracking Application)

User Management

Send Messages and Notifications

Error Logs from mobile devices (for diagnosis purpose)

Other Admin functions

Mobile Application Over the Air upgrades Management

Check the status of other system component and SNMP alarms

Figure 1.12: Gateway Management Application

The application can be accessed by multiple users at the same time over internet using a web browser. It is highly secure using HTTPS, password protected and accessible from only allowed IP address ranges. The application was developed in C# using ASP.NET 2.0. It also uses AJAX.NET for real-time data status update without refreshing the web page. The application interacts with the central SQL Server 2008 database and also utilises the gateway services.

Commercial benefits

The gateway services provided a unified communication and integration layer by combining T@lecom's legacy web services with the new WCF services as part of the SOA platform. It was a cost effective approach and avoided the re-development of legacy SOAP web services.

The management application enabled access over internet (from trusted source) allowing technical administrators to remotely manage the gateway. This greatly improved T@lecom's support service especially during out of normal hours where administrators can even provide support from their homes in critical situations.

Relevance to the research

The development of the Gateway Services and Management web application contributed to the research in following ways:

- Various components, libraries, code (classes) of both application and services were originated or used in ITS@CU implementation
- Acquired valuable knowledge/experience in various tools/technologies especially the SOA based implementation which was also useful in the development of ITS@CU platform

Appendix B: Other research implementations & contributions

1. Research publications

The author has also published the following scientific papers based on various outcomes of the research:

- “Detecting traffic incidents based on traffic patterns and vehicle behaviours using GPS”, In Proceedings of the 18th International Conference on Systems Engineering (ISCE06), Coventry, United Kingdom, 5-7th September 2006, pp. 201 – 206 by Kamran, S., Black, J. and Haas, O. C. L. (2006).
- "A Multilevel Traffic Incidents Detection Approach: Identifying Traffic Patterns and Vehicle Behaviours using real-time GPS data", In proceedings of the Intelligent Vehicles Symposium, IEEE, Istanbul, Turkey, 13-15th June 2007, pp. 912 – 917 by Kamran, S. and Haas, O.C.L, (2007).
- “Emergency response time optimisation using real-time traffic information”, In proceedings of the 7th International Conference on Transport Systems Telematics (TST’07), Katowice-Ustroń, Poland, 17-19th October 2007 by Jaskulowski, M., Kamran, S. and Haas, O.C.L, (2007).
- “Semantic Agent-based Controls for SOA enabled ITS”, In proceedings of the 14th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2011), The George Washington University, Washington, DC, USA, October 5-7, 2011 by Kamran, S. and Haas, O.C.L, (2011)

2. Student projects/dissertations

During the research, the author has worked with other undergraduate and postgraduate students at CTAC as an external supervisor, and also specified various dissertations related to this research. These projects benefited both the students and this research; the students were given the opportunity to become part of a commercial research project and gain industrial exposure, and the research benefited in a variety of ways from the results of the students' work. Following are some of the completed dissertations:

- Agent technology in Intelligent Transportation Systems (2008) by Abdelouahab Benssassi
- Agent based communication and control in Intelligent Transportation Systems (2008) by Stanislaw Kardach
- Agent Communication Systems (2008) by Bertrand Stivalet
- Emergency response Time optimisation using Traffic information (2007) by Mirosław Jaskulowski
- Mobile Computing for Intelligent Transportation System (2007) by Kamil Baczkowicz

3. Patent idea

The author has also instigated a patent of an idea in conjunction with T@lecom, based on the Personal Area Network. The patent idea is to develop a Bluetooth based proximity application for security devices/dongles, mobile phones and other such devices/appliances. The idea originated from the ad-hoc Bluetooth based Personalised Area Network (PAN) application development for V2V communication described in *chapter 7*. Currently the patent is in the final stages of filing.

4. Other Research implementations

This research project involved wide ranging experiments for analysing various new methods, concepts and technologies relevant to the subject area. As a result of these R&D efforts, different research applications and simulation systems were developed and evaluated, and also published as research papers.

Some of the research implementations relevant to this research are outlined below:

4.1. GPS based incident detection system

In the early stage of the research the author investigated different incident detection approaches in order to develop an effective incident detection system as part of the ITS@CU platform. As a result of these investigations, the author developed an experimental Incident Detection System based on a new approach for detecting traffic incidents causing congestion on major roads. This approach incorporated algorithms to detect unusual traffic patterns and vehicle behaviours on different road segments by utilising the real-time GPS data obtained from vehicles.

Technical Description

The system comprised of various components i.e. In-Vehicle mobile application, server controls, SQL Server 2005 Databases, and a web based mapping front end application (See *figure 1.1*)

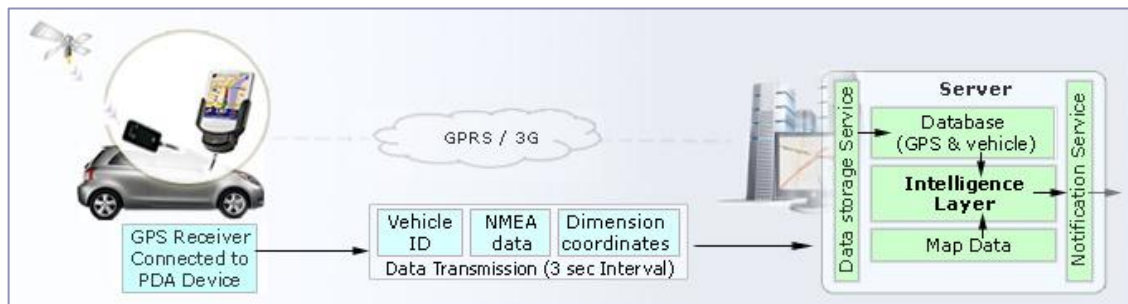


Figure 1.1: Incident Detection System Overview

The PDA application interprets the GPS signals every 5 seconds and converts it from NMEA to WSG84 format which is a standard that can be used with the Microsoft Virtual Earth mapping service. The PDA application uses GPRS/3G to communicate with the server communication service layer (SOAP based web service hosted on an IIS enabled server), transfer the real time GPS data and receive server alerts (advance driver warnings).

The intelligence layer is a Windows Service application and an important part of server controls. It uses multi-level detection algorithms for detecting traffic incidents which are causing congestion on major roads. The algorithms detect unusual traffic patterns and vehicle behaviours on different road segments by utilising the real-time GPS data obtained from vehicles.

The incident detection process involves two phases:

Phase 1: Identifies road segments where an abnormal traffic pattern is observed and further divides the ‘abnormal segments’ into smaller segments in order to isolate the potential incident area;

Phase 2: Performs a hierarchical analysis of the vehicles’ GPS data, using pre-defined rules to detect any occurrence of abnormal behaviour within the ‘abnormal’ road section identified in phase 1.

PHASE 1 ALGORITHM	PHASE 2 ALGORITHM
Analyse traffic patterns on road R 1: Start Segmentation process (R, Type, Boundary coordinates) Switch Conditions (Time, Date and Weather) // cases predefined Return Segments details //Segment Length, boundary coordinates, Normal_Avg_Speed Classify upstream downstream 2: For Each Segment (or sub-segments) S in R Calculate $\text{Current_Avg_Speed}(S) = \frac{1}{N} \sum_{i=0}^N \text{vehicle_Speed}(S)$ // where N=number of cars in segment S 3: IF Current_Avg_Speed (S) << Normal_Avg_Speed (S) Mark Segment S; 4: Do While Segment S with lowest Current_Avg_Speed is identified 5: While Current_Avg_Speed (S) ≥ Current_Avg_Speed (S±1) S=S±1; Else Process Segment S; 6: Get Current_Avg_Speed (S±1) IF (Current_Avg_Speed (S+1) >> Current_Avg_Speed (S)) && (Normal_stoppage_pts (S) == 0) Mark Segment S as possible incident location 7: Start Sub-Segmentation process (S, Boundary coordinates) Segment_Length = Segment_Length / 10; Repeat Step 2 to 6 // for all the sub-segments 8: Wait for interval T IF no change in Current_Avg_Speed (S) Execute Phase 2 on S	Identify vehicle behaviour in sub-segment S 1: For 1:N //N = number of cars IF Speed (N) << Normal_average_speed Speed (N) ≤ 0 Direction (N) not within the normal range of the flow Mark Vehicle_N 2: IF (Normal_stoppage_pts (S) == 0) Goto 3 Else Goto 4 3: Wait for interval T1 IF no change Goto 4 4: For Each Vehicle_N in S Analyse GPS Data (Vehicle_N) // abnormal deceleration, direction change, collision etc. (See 2.4) 5: Wait for interval T2 IF no change Alert and return (Location coordinates, Vehicles_List)

Figure 1.2: Incident Detection Algorithms

The strength of such an approach lies in isolating road segments sequentially and then analysing vehicle data specific to the identified road segment. In this way, the processing of vast data is avoided which is an essential requirement for the better performance of such complex systems.

The approach was demonstrated using simulated traffic data and the results were published in a research paper for the “IEEE Intelligent Vehicle Symposium 2007” research conference. The following are the summary of the results:

DETECTION RATE (DR)					FALSE ALARM RATE (FAR)				
Phase	Vehicles Per Seg. (on different time interval)	Seg.	Road Type	DR % (Average)	Phase	Vehicles Per Seg.	Segments	Road Type	FAR %
1	10 - 20	10	Motorway	78.70	1	10 - 20	10	Motorway	1.18
1	20 - 80	10	Motorway	75.30	1	20 - 80	10	Motorway	1.23
1	100 - 200	10	Motorway	71	1	100 - 200	10	Motorway	1.37
1	100 - 200	20	Motorway	71.60	1	100 - 200	20	Motorway	1.38
2	10 - 20	10	Motorway	62.70	2	10 - 20	10	Motorway	2.16
2	20 - 80	10	Motorway	59.30	2	20 - 80	10	Motorway	2.28
2	100 - 200	10	Motorway	53.60	2	100 - 200	10	Motorway	2.31
2	100 - 200	20	Motorway	52.20	2	100 - 200	20	Motorway	2.32

DR = Detected accidents divided by actual accidents

FAR = Number of incident free interval with false incidents alarms divided by the total number of incident free intervals

MEAN TIME-TO-DETECT (MTTD)				
Phase	Vehicles Per Seg.	Segments	Road Type	Avg. MTTD (hh:mm:ss.ms)
1	10 - 20	10	Motorway	00:00:2:20
1	20 - 80	10	Motorway	00:00:2:20
1	100 - 200	10	Motorway	00:00:2:20
1	100 - 200	20	Motorway	00:00:2:30
2	10 - 20	10	Motorway	00:03:3:40
2	20 - 80	10	Motorway	00:03:3:40
2	100 - 200	10	Motorway	00:03:3:40
2	100 - 200	20	Motorway	00:03:3:50

MTTD = Difference between the time of accident occurrence and the time of accident detected

Figure 1.3: Results of the incident detection system in terms of DR, FAR and MTTD

See “Appendix B (IEEE IV 07)” research paper for further details and results. The complete source code of the applications including instructions can be found in the attached CD, Appendix J.

The following are some of the screen shots of applications developed as part of the Incident Detection System.



Figure 1.4: PDA mobile application used in vehicles

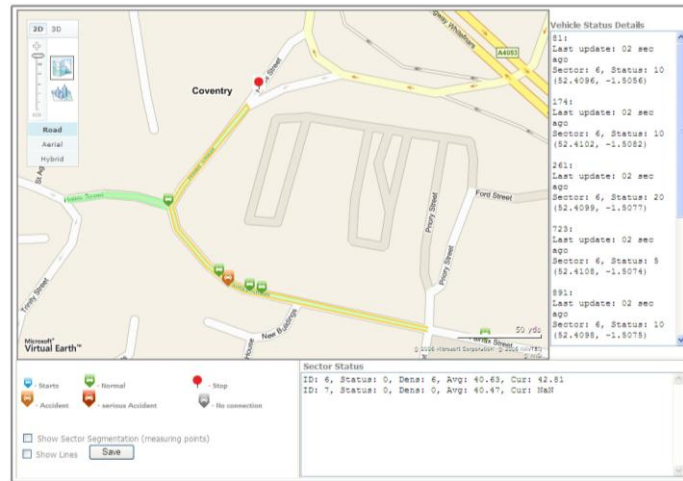


Figure 1.5: Web based mapping interface showing the vehicles and accidents

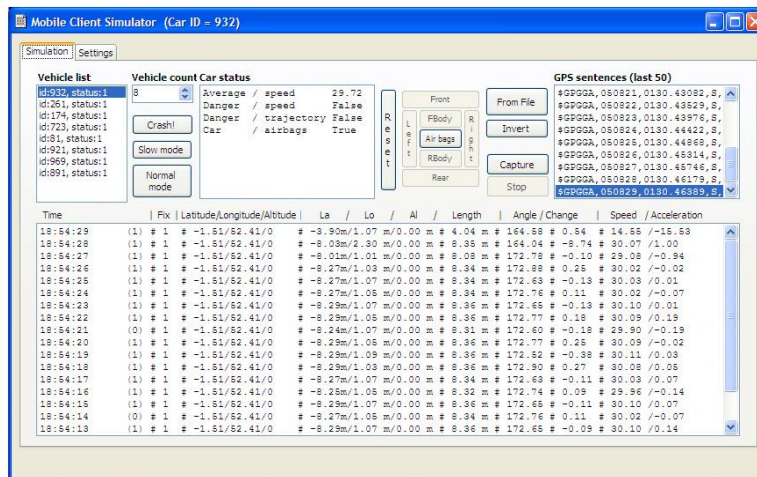


Figure 1.6: Windows application for simulating various real time incident scenarios by generating bulk GPS data

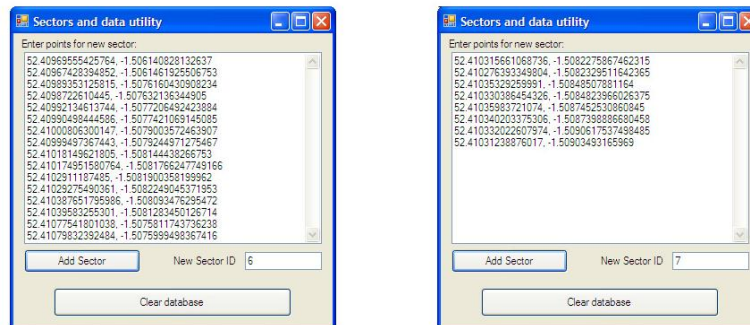


Figure 1.7: Road segmentation utility for creating custom road segments and grids

4.2. Simulation system using Personal Area Network (PAN)

In this research project, various Vehicle to Vehicle (V2V) technologies were analysed as part of the ITS@CU platform development. As a result an experimental mobile application was developed for simulating Vehicle Controls forming ad-hoc Personal Area Networks (PAN) using Bluetooth. The application was developed using the Mobile Application Development Framework (described in *chapter 7, section 7.4*) and it was predominately used for testing the communications between vehicle-to-vehicle and vehicle-to-traffic-controls.

The application uses Bluetooth listener classes (based on 32feet.NET libraries) and XML for describing the communication ontologies. Using this PDA application, a number of Vehicles can be set up to form ad-hoc networks, however the PDA application only allows for one Controller at a time. So in order to test ad-hoc PAN based Multi-vehicle interaction and their behaviour, multiple PDA devices were required with Bluetooth enabled and paired up. The author had tested up to 7 PDAs at the same time for simulating various traffic scenarios.



Figure 1.8: Ad-hoc V2V PAN Application

Left: Application initial setup; **Centre:** Bluetooth setup and ad-hoc vehicle/device in range info;

Right: Customised V2V Messages for simulation/Test purposes

The complete application source code including the instructions can be found in the attached CD, *Appendix J*.

4.3. Guidance system for emergency response vehicles

This was a research system providing real-time route guidance for emergency response units such as ambulances. The research was conducted in association with CTAC research group, Coventry University. It involved the development of an experimental application to demonstrate a solution supporting emergency vehicle drivers with optimal real-time route guidance using on board navigation software. It provided intelligent navigation with real-time traffic congestion information evaluated for the calculation of the fastest path to the destination. The application uses TomTom for alternative route generation however the application receives congestion alerts from a server control application using web services over 3G/GPRS.

The results were published in a research paper for “Transport System Telematics Conference 2007”. See “*Appendix B (TST 07)*” research paper and poster for further details and results.

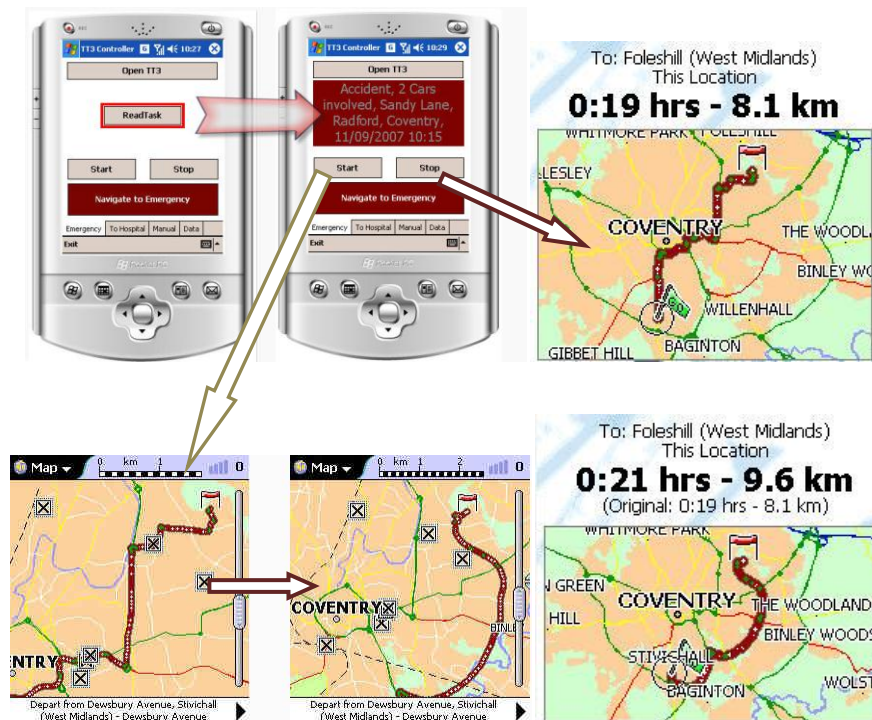


Figure 1.9: Alternative route generation by TomTom controlled by the mobile application based on real-time congestion alerts from server

Appendix C: Intelligent Transportation Systems (ITS)

1. ITS functional and research areas

The ITS based systems and applications are further divided into following areas:

- Advanced Traffic Management Systems (ATMS)
- Advanced Traveller Information Systems (ATIS)
- Advanced Vehicle Control Systems (AVCS)
- Commercial Vehicle Operations (CVO)
- Advanced Public Transport Systems (APTS)
- Advanced Rural Transportation Systems (ARTS)

The ITS functional areas categories by service type:

ITS Category	Service Areas
Travel and Transportation Management:	<ul style="list-style-type: none"> • En-route driver information: Systems and services to provide up to date information to vehicle drivers. • Route guidance: Provides directions for optimal route to the driver's destination. • Smart Traffic control: Effective management of the traffic flow. • Incident management: helps quicker incident detection and notifying the response authorities. • Highway-Rail Intersection: helps avoid accidents at railway crossings. • Location based services: Provides a location based services to the travellers such as Motorway services, Fuel/gas stations, hotels etc. • Emissions Quality Control: Air quality testing and mitigation strategies.
Public Transportation Operations:	<ul style="list-style-type: none"> • Public transportation management: automates operations, planning and management functions of public transportation systems. • Public Transportation Information services: Provides up to date information to travellers such as timetables, journey planners, station information etc.
Travel Demand Management	<ul style="list-style-type: none"> • Advance Travel Information: Systems and services to provide information to travellers for selecting the best transportation mode, departure time and route. • Demand management and operations: Supports policies and regulations in order to mitigate the environmental and social impacts of traffic congestion.
Commercial Vehicle Operations	<ul style="list-style-type: none"> • Mobile Fleet Management: Effective fleet management tools and techniques such as automated dispatching, satellite tracking and optimal routing for multiple dispatch locations, and better communication systems to enable commercial vehicle users with their despatch centres. • Commercial vehicle electronic clearance: Automated vehicle identification and secure clearance to minimise vehicle stoppage • Hazardous material incident response: provides immediate description of hazardous materials to emergency responders.
Emergency Management	<ul style="list-style-type: none"> • Emergency notification: Systems to provide immediate notification of an incident with its location, nature and any other information to the response authorities.

	<ul style="list-style-type: none"> • Emergency vehicle management: Helps rescue and response vehicles to guide and assist to the incident.
Advanced Vehicle Control and Safety Systems	<ul style="list-style-type: none"> • Collision Avoidance: Systems to prevent vehicle to collide with another vehicle or road side objects • Vision enhancement: Systems to enhance the visibility of the driver to see and assess the road-side objects and hazardous situations. • Safety Warnings/Alert systems: Advance driver information about the road condition, vehicle as well as driver. • Pre-Accident restraints: Activation of passenger safety systems at the right time before accidents to limit or avoid passenger's physical injuries.
Electronic Payment	<ul style="list-style-type: none"> • Electronic payment: Services to allow travellers to pay easily and safely for transportation services electronically (tolls, transit fares, and parking)

2. Major ITS Systems & organisation

Some of the major ITS related systems include WAIMSS, RHODES, DATAGRID II, eDAPTS, MADARP, COMPASS (Freeway Traffic Management System), MIDAS (Motorway Incident Detection and Automatic Signalling), PRIME, TransVista, SCATS, SCOOT, OPAC, Active Traffic Management (near Birmingham on the M42, UK)

Some major In-Vehicle ITS systems are OnStart, BMW ConnectedDrive, SARA (Short-range Automotive Radar), PRE-SAFE, DSRC (Dedicated Short Range Communications, IVCS (Inter-Vehicle Communication System)

Institutions and research groups dealing with ITS

United Kingdom

- ITS United Kingdom, the intelligent transport society for the UK (<http://www.its-uk.org.uk>)
- Department of Transport, UK (<http://www.dft.gov.uk>)
- InnovITS research group (<http://www.innovits.com>)
- IBEC (<http://www.ibec-its.org>)

Europe

- ERTICO – ITS Europe (<http://www.ertico.com/en/home.htm>)
- Trans-European Network for Transport (<http://www.ten-t.com>)
- VIKING, for northern Europe (<http://www.viking.ten-t.com/VikingExtern/Index.htm>)
- CENTRICO (<http://www.centrico.ten-t.com>)

World

- Intelligent Transportation Society of America (<http://www.itsa.org>)
- The Centre of Transportation Analysis (<http://www.cta.ornl.gov/cta>)
- U.S. Department of Transportation (<http://www.its.dot.gov>)
- Intelligent Transportation Systems Program on MIT (<http://web.mit.edu/its>)
- Intelligent Transportation Systems Society (<http://www.ewh.ieee.org/tc/its>)
- ITS Japan (<http://www.its-jp.org/english>) , ITS Australia (<http://www.its-australia.com.au>)

3. Data fusion technologies & projects

Following are some of the ITS based projects using different data fusion techniques:

Project	Technique(s)	Purpose
ADVANCE (Kirson et al.)	Kalman filter	Forecasts future traffic conditions
	Neural network	Pattern-matches current traffic situations with historical situations
	Expert system	Identifies abnormal traffic conditions
	Fuzzy logic	Permits traffic conditions to be described with qualitative measure rather than simple "yes-no" responses
PROMETHEUS (Behringer et al.) (Martinez et al.)	Kalman filter	Constructs 4-D position estimates for autonomous driving
	Expert system	Decomposes a driving task into independent subtasks
	Neural network	Allocates one neural net for each driving subtask
Brainmaker	Neural network	Pattern-matches current traffic situations with historical situations
IGHLC (Niehaus, Stengel)	Kalman filter	Determines vehicle position
	Bayesian	Deals with traffic uncertainty
	Expert system	Models concept of Worst-Case Decision Making
Pathfinder (Sumner)	Fuzzy logic	Permits traffic conditions to be described with qualitative measure rather than simple "yes-no" responses
TravTek (Sumner)	Fuzzy logic	Permits traffic conditions to be described with qualitative measure rather than simple "yes-no" responses
DRIVE (Martinez et al.)	Expert system	Decomposes a driving task into independent subtasks
	Neural network	Allocates one neural net for each driving subtask
PRODYN (Kessaci et al.)	Kalman filter	Estimates traffic-turning movements
	Bayesian	Estimates traffic-state variables, e.g., queues and
Application to AGVs: Autonomous Guided Vehicles (Harris & Read)	DSER	Determines state of AGV and outside world

ITS data fusion projects

4. Traffic data services providers

Various agencies and sources are involved in the data collection and processing such as:

- **Traffic Infrastructure operators:** Responsible for monitoring traffic networks such as traffic lights management and motorway information displays etc.
- **Traffic Information services:** Local authorities providing off-line data for statistical and planning purposes for a certain area usually in cooperation with infrastructure operators.
- **Private Service:** Collating data from various sources to provide commercial/user services for example traffic information feeds for navigation application, fleet route planning and public transport scheduling or selling traffic information tailored to the customer's needs (McDonald and Li, 2006).

Electronic Local Government Information Network (ELGIN)	A website providing an interactive map and XML feed of current and planned road works plus diversionary routes. Data is updated on a daily basis and made available via DATEX II and SDEP protocols. As it can be observed on the map on the following page, the data is grouped in High Medium and Low impact on traffic categories. There are also current access restrictions displayed and future road works.
BBC Traffic and Travel Information	Service providing free of charge data on road traffic and public transport. The information is encoded as TPEG-ML and available via the Internet. Events are referenced using TPEG-locML and refreshed every 10 minutes.
MAT'TISSE	A service designed to cover traffic information from Midlands's area. For data exchange, service uses UTM v1.8 protocol, locations are referenced by Ordnance Survey Grid Reference (OSGR Eastings and Northings, arises historically from the creation of 2D paper maps). Data is updated in real time and available via the Internet.
National Traffic Control Centre Data Services	This service covers 6500 km of motorways and major A-roads which is 2% of all roads in England (350'000km in total (SABRE, 2007)). Information on planned and unplanned road events which influence on road is more than 15 minutes is available via XML service.
TrafficMaster	Trafficmaster specialises in monitoring traffic and runs various data mining projects to picture "congestion hot spots" or local effects of school traffic. Smartnav is Trafficmaster's satellite navigation system making use of data collected by 7500 fixed infrared and CCTV sensors covering 8000 miles of main UK's roads along with 50000 intelligent vehicle probes. Smartnav utilises real-time traffic alerts (received when speed on motorways and A-roads fall below 30mph) and average speeds database collated from 50 billion data records.

Traffic Information publishers both from private and public sector operating in UK.

5. Traffic Simulation

There are different type of models employed by traffic simulators based on traffic control strategies, area of coverage and level. Some of the well-known traffic simulators include:

VISUM: Flexible and comprehensive simulation system for transportation planning, travel demand modelling and network data management. VISUM focuses on multimodal analysis by integrating all relevant modes of transportation such as car, passenger, truck, bus, train, pedestrians and cyclists into one consistent network model. It provides a variety of assignment procedures and four stage modelling components which include trip-end and activity based approaches.

VISSIM: It is a widely used microscopic simulation tool for multi-modal traffic flow modelling. It provides high level of detail, able to accurately simulate urban and highway traffic, including pedestrians, cyclists and motorized vehicles.

SOUND: It is based on the block density method (BDM) developed by the University of Tokyo in 1971, which is aimed at the online congestion estimation of near future. SOUND deals each vehicles behaviour to save a computational time. It can be applied to a large road network for analysing congestion situation with high accuracy.

MITSIM: It is a microscopic simulator developed by Massachusetts Institute of Technology in 1996, with the ability to combine an urban network and a highway network simulation. MITSIM allows evaluating the advanced traffic management systems and the driver behaviours. It can handle highways and urban roads at the same time, can adapt changes in traffic.

SATURN: SATURN (Simulation and Assignment of Traffic in Urban Road Network) was developed by Leeds University in 1978. It describes the traffic flow as fluid and analyses the route choice. It can also model the traffic flow in detail at intersection level.

NETSIM: The NETSIM (NETwork SIMulator) is a microscopic traffic flow simulator which allows evaluating various Travel Demand Management (TDM) strategies in urban road. NETSIM outputs traffic flow rate, travel time, delay time and evaluates emission of gases and fuel consumptions. It effectively deals multiple calculations by using batch processing.

BOX Model: It was developed by Kyoto University in 1990, primarily to evaluate traffic lights control strategies and route guidance. Its main feature is the effective calculation of a link flow.

CORISM: It is a comprehensive microscopic traffic simulation used widely over last 30 years. It is applicable to surface streets, freeways, and integrated networks with a complete selection of control devices (i.e., stop sign, traffic signals, and ramp metering). It simulates traffic and traffic control systems using commonly accepted vehicle and driver behaviour models. CORSIM combines two of the most widely used traffic simulation models, NETSIM for surface streets, and FRESIM for freeways/Motorway (FHWA).

6. Mapping traffic network

Some of the well-known formats and mapping standards are discussed below:

Geographic Data File (GDF)

GDF developed by Comité Européen de Normalisation (CEN) is a widely accepted standard for describing and exchanging road networks and road-related data in form of navigable databases. GDF exchange format specifies a sequential ASCII file for off-line data transfer in one-way communication. GDF can be used in vehicle navigation, dynamic route guidance, location-based services, fleet management, public transport and road administration (Essen and Hiestermann 2005). It specifies the conceptual and logical data model, and the exchange format for geographic databases for ITS applications. It includes a specification of potential contents of such databases (features, attributes and relationships), a specification of how these contents shall be represented, and of how relevant information about the database itself can be specified (Innocenti et.al 200X; Essen and Hiestermann 2005).

The overall data model of GDF contains three entities or models, Feature, Attribute and Relationship.

Feature Model:

Feature is the main entity which represents real world geographic object such as roads or buildings. A feature belongs to exactly one 'feature class' and exactly one 'feature theme' that are uniquely referenced by a name and a code. GDF features are defined by the feature catalogue/themes: Roads and ferries, administrative areas (cities, counties, countries, etc.), named areas (postal areas, police district, etc.), (parks, airports, olive grove, etc.), Waterways, Public transport or Custom features (ISO 2004; REWERSE 2005, Innocenti et.al 200X).

The different objects together making up a GDF are conceptually divided over three different levels.

Level 0: Defines the basic graphical building blocks of the map, which are Nodes, Edges and Faces. They together contain the geometrical and topological information of the Features which refer to them.

- A 'Node' represents a zero-dimensional location on the earth surface. It constitutes a zero-dimensional building block.
- An 'Edge' represents a one-dimensional location on the earth surface. It constitutes a one-dimensional building block. An edge is bounded by a start and an end node.
- A 'Face' represents a two-dimensional location on the earth surface. It constitutes a two-dimensional building block.

In alternative non-explicit topological terms, these basic building blocks are objects: Dots, Polylines and Polygons.

- A Dot is a zero dimensional location on the earth surface.
- A Polyline is a one-dimensional location on the earth surface.
- A Polygon is a two-dimensional location on the earth surface by means of a representation of the objects boundary.

In order to unambiguously define a Polygon, it is necessary that it is defined relative to the order of the coordinates of the boundary. The Polygon will always be located on the right side of the sequence of coordinates. It is allowed that a Polygon is defined by more than one boundary. In this way, enclaves and exclaves are defined according to the same principle as stated above. The individual boundaries will always have identical start and end coordinates.

The Features are divided over level-1 and level-2 depending on whether they are simple (level-1) or complex (level-2).

Level 1: Defines simple features like Points, Lines and Areas. A road network on this level is represented by junctions (Point Feature) and roads (Line Feature). Area features can be used for address areas, counties, etc. As features can have attributes such as number of lanes, road name, speed limit, turn limitation, this level is used for en-route assistance and also for location on the map. Below an example of Features from Roads & Ferries theme:

Level 2: defines complex Features composed of one or more simple Features. In this level a road element and a junction can be aggregated. This level is used to calculate the shortest route. Road and Ferry theme contains classes:

Attribute Model:

Characteristics of features which are independent of other features are modelled as attributes. It defines over 200 simple or composite/complex (more than one simple) characteristics of Features and possibility of their relationships. Attributes are grouped so they apply to specific Features but one Feature might have any number of Attributes. They are of a certain attribute type identified by the name and the code. An attribute may be an aggregation of other attributes. Such an attribute is called a 'composite attribute'. A composite attribute consists of a number of sub-attributes. GDF attributes are defined by the attribute catalogue, e.g. 'street names', 'house numbering', 'time domains', 'official language', and 'speed restrictions' (Larbo 2000).

Relationship Model:

Some information related to real world objects needs to be modelled in the form of a relationship between features and links between levels. Examples of such relations are prohibited and permitted manoeuvres, description of crossings or services along a road etc. The GDF Standard defines and describes all Relationships (Larbo 2000). The content of the GDF file with map database is build of 80 ASCII character per line records related to by pointers. The Line Feature contains references to other records. GDF databases used for route calculation require updating and one method to tackle the dynamism of road network parameters is to integrate pre-defined Traffic Messaging Chanel (TMC) locations database into a map. Such locations can be later referenced to enable traffic messaging. TMC defined as a specific point identified as a landmark are embedded into the map. In the GDF format locations are stored as a database TMC locations are present in a Level 1 or 2 as attributes identified with letters RD followed by TMC unique ID.

Geography Markup Language (GML)

GML is an XML grammar written in XML Schema for the description of application schemas as well as the transport and storage of geographic information (GML spec). The key concepts in GML for modelling the geographic information or mapping are based on earlier ISO 19100 Standards and the OpenGIS. GML focuses on the representation of the geographic data content on a map and does not actually used for creating maps. Like GDF, GML is also based on the abstract model of geography which describes the world in terms of geographic entities called features. A feature is a list of properties and geometries; properties have the usual name, type, value description, and geometries are composed of basic geometry building blocks such as points, lines, curves, surfaces and polygons. The new GML specification also supports 3D geometry and topological relationships between features. The type definition of the feature determines the number of properties together with their names and types. A feature collection is a collection of features that may itself be regarded as a feature; as a consequence a feature collection has a feature type and thus may have distinct properties of its own, in addition to the features it contains (Morris and Petry, 2006).

In GML encoding, feature can be quite complex i.e. composed of other features in various layers. E.g. A feature train station consists of other complex features such as taxi stands, parking and platforms. The geometry of a geographic feature can also be composed of many geometry elements i.e. consist of a mix of geometry types including points, line strings and polygons. GML also encodes spatial reference systems, which means referencing the geographic features to the earth's surface or to some structure related to the earth's surface. The current version of GML incorporates an earth based spatial reference system which is extensible and which incorporates the main projection and geocentric reference frames in use today.

There are already a host of encoding standards for geographic information such as COGIF, MDIFF, SAIF, DLG and SDTS. As compared to other encoding standards, GML is a XML based text which can be universally exchanged between systems due to XML which is open, vendor neutral, robust, transformable and easily editable and upgradable.

7. Incident detection algorithms

A number of incident detection algorithms have been developed over the years and with a variety of theoretical approaches. These AID algorithms can be classified in four categories:

Model based detection algorithms: In model based detection algorithms, traffic flow models are derived and validated using historical records. These models are usually non-linear and operate at the macroscopic level. They can be implemented from past traffic information, using dynamic state space techniques to estimate the state of the traffic (in terms of density and flow) as well as perform additional observations such as on-ramp entrances to a particular road or segment length and capacity. These models can then be exploited, to predict the evolution of traffic pattern. If the traffic differs significantly, then it may be that an incident has occurred. A standard alternative to model complex non-linear behaviour is neural networks (NN). NN can be trained on past data to recognise traffic flow patterns and so recognise states associated with the presence or absence of incidents. Whilst effective, NN approach can be difficult to train and convergence to a solution can be slow. Understanding what a NN model means is difficult and a large amount of historical data are required for training purposes. Fuzzy logic has been used to overcome issues associated with scarce data and capture knowledge based on expert experience. Fuzzy algorithms have used the idea of a fuzzy boundary and the change in occupancy or relationship between speed and density between neighbouring detector stations to identify traffic incidents.

Image-based algorithms: Mainly used at the micro or local level, detect and verify incidents from image sequences. Video image processing (VIP) is fairly new technology with diverse applications and promising potential. Understanding its capabilities and limitations, and supplementing it with other appropriate traffic sensors could prove to be a very efficient tool for incident detection and management.

It is capable of obtaining lane occupancy, volume counts, speed data, density, headway, vehicle classification (via vehicle length), and queue length, as well as other traffic characteristics. It also may be used to monitor wrong way vehicles and lane changes, and to provide a real-time video screen for incident monitoring (Awadallah and Habesch 1998).

Logical commands (such as 'and' or 'greater than') may be used in algorithm analysis programs to predict incident detection from various traffic parameters. This may be obtained from commands such as vehicle presence for n or more seconds, vehicle speeds less than n , and/or queue length exceeding n .

The camera's field-of-view (FOV) is the most important element for data and traffic parameter representation, and the larger the FOV, the larger the percentage of error, and the larger the percentage of false alarm for incident detection (Awadallah and Habesch 1998). Variation of object and background lighting, particularly at night, heavy rain, fog, sand storm and during lighting storms, requires special filtering attention.

Prediction algorithms: are usually applied in situations where traffic forecasting is required. Historical traffic data is used to analyze the traffic-flow parameters using statistical forecasting techniques and compares the predictions to the actual flow to identify incidents. In this approach, the detection algorithms are both prediction and method based.

Pattern recognition algorithms: In traffic pattern recognition algorithms, the aim is to recognize and discriminate between different traffic patterns using data from detector stations for example, monitoring the upstream and downstream occupancy using loop detector stations on a freeway or motorway. The expected state is when occupancy increases upstream, but decreases downstream. An incident is detected when upstream and downstream occupancy passes predefined thresholds.

Approach	Description	Example Algorithms
Pattern recognition	Recognise and differentiate unusual traffic flow patterns from normal by monitoring the occupancy levels	California algorithm 7/8 GPS based
Image Recognition	Analyse the real-time images of the traffic flow to identify the congestion and blocked lanes.	

statistical models:	Analysis data statistic from loop detectors/sensors to establish the abnormal condition	Standard Normal Deviation (SND), Bayesian Time series and filtering
Theoretical	Monitors the characteristics of traffic flow such as volume-occupancy	McMaster
Artificial intelligence/Neural Networks	AI and NN based concepts by using the traffic flow data (volume, speed and density)	

Following is a summary of the comparative performance of different algorithms is given below.

Algorithm		Detection Rate [%]	False Alarm Rate [%]	Average Detection Time [minutes]
California	Basic	82	1.73	0.85
	California #7	67	0.134	2.91
	California #8	68	0.177	3.04
	APID	86	0.05	2.5
Standard Normal Deviate		92	1.3	1.1
Bayesian		100	0	3.9
Time Series ARIMA		100	1.5	0.4
Exponential Smoothing		92	1.87	0.7
Low-Pass Filter		80	0.3	4.0
Modified McMaster		68	0.0018	2.2
Neural Networks	MLF	89	0.01	0.96
	PNN	89	0.012	0.9
Fuzzy Set		Good	Good	Up to 3 minutes quicker than conventional algorithms
Wave Analysis		Good	Good	Good
Dutch		Good	Poor	Good
Monica		Poor	Good	Good
Low-Volume Algorithm		49-78	Volume < 400vph: 1 per 7 hrs Volume 900-1000 vph: 1 per 2 hrs	N/A
Logit Based	x	96.3	5.3	Good

Reported Algorithm Performance Summary (Ozbay, 1999)

Appendix D: Agent Oriented Technologies Review

1. Multi-Agent communication using Agent Communication Language (ACL)

ACL is based on KQML. It is an agent based communication language according to FIPA standard, often referred as FIPA ACL. A FIPA message consists of a sender, a receiver, contents and a performative. The performative is used to classify the message into meaningful low-level intentions. *There are 22 performatives classified into the following groups:*

Performative	Description
<i>Confirm</i>	This message allows the Sender of the message to confirm the truth of the content to the recipient
<i>Disconfirm</i>	This message is sent whenever a sender agent wants to confirm the misbelieve of some statement
<i>Inform</i>	This message is sent whenever an agent wants to share information
<i>Inform-if</i>	This message is sent to verify if a statement is true or false. Typically as the contents within a request message
<i>Inform-ref</i>	Similar to the inform-if message with the difference being that it is the value of an expression that is being asked for

Table D1.1: Passing information

Performative	Description
<i>Cancel</i>	This is sent whenever an agent wants to cancel a previously submitted request
<i>Query-if</i>	This message allows for agents to query if something is true or not
<i>Query-ref</i>	This message is used to ask for values of expressions
<i>Subscribe</i>	Sent whenever an agent wants to be notified when some event occurs

Table D1.2: Requesting information

Performative	Description
<i>Failure</i>	Message sent to indicate that the agent failed to perform some action
<i>Not-understood</i>	Sent whenever an agent doesn't understand the message of another agent

Table D1.3: Error handling

Performative	Description
<i>Agree</i>	Indicates that the agent agrees to perform some action
<i>Cancel</i>	This is sent whenever an agent wants to cancel a previously submitted request
<i>Propagate</i>	Message sent when an agent wants another agent to propagate a message to other agents
<i>Proxy</i>	Used for forwarding of messages
<i>Refuse</i>	This message is used to indicate that the agent refuses to perform some action
<i>Request</i>	Request for some agent to perform some action
<i>Request-when</i>	The same as request but only if some statement is true
<i>Request whenever</i>	Similar to request-when with the difference that the requested action should be performed whenever some statement is true

Table D1.4: Performing Actions

Performative	Description
<i>Accept-proposal</i>	Allows for an agent to state that it accepts another agents proposal
<i>CFP</i>	Call For Proposals is used to initiate negotiations
<i>Propose</i>	This message is sent to make a proposal to another agent
<i>Reject-proposal</i>	This message rejects a suggested proposal

Table D1.5: - Negotiation

Content of ACL Messages

According to FIPA the content of an ACL message can be encoded in any content language. A content language must be able to express propositions, objects and actions. No other properties are required, though any given content language may be much more expressive than this. More specifically, the content of a message must express the data type of the action: propositions for inform, actions for request, etc. (FIPA, 2003). In this context, a proposition can be a sentence, e.g. in predicate-logic, which can be true or false. An object represents an abstract or a concrete entity, which does not necessary appear in an object-orientated languages. An action is considered as an activity, carried out by an agent. Possible candidate content languages are KIF, Semantic Language (SL), Prolog and eXtended Markup Language (XML).

2. Agent oriented Tools and Technologies

Agent oriented computing is relatively less established area, and the tools and technologies involved still lack standards. However, various standards are being discussion (Tweedale et al. 2007), and agent technology is getting serious attention by major software vendors. Currently, most of the commercial products/toolkits are proprietary and quite specific in nature. Following are some of the tools and technologies currently available.

Name	Type	Sector	By
JADE	Support software / Platform	Commercial	TILAB
JADE / LEAP	Distributed Agent Platform	Commercial	Motorola, ADAC, Broadcom, BT, TILabs Uni Parma, Siemens
RePast	MAS-based modelling/simulation tool	Academic	University of Chicago
NetLogo	Language or environment for MAS development	Academic	North-western Uni
StarLogo	Language or environment for MAS development	Academic	MIT MEDIA lab
Voyager	Support software	Commercial	Recursion Software, Inc.
JACK	Language and Environment for MAS development	Commercial	Agent Oriented Software Group
3APL	Language and environment for MAS development	Academic	Utrecht University
CORMAS	Language or environment for MAS development	Commercial	CIRAD
Bee-gent	Language or environment for MAS development	Commercial	Toshiba Corporation
ADK	Language or environment for MAS development	Commercial	Tryllian
AgentSheets	Language or environment for MAS development	Commercial	AgentSheets Inc.
CABLE	Language or environment for MAS development	Commercial	Logica UK Ltd
Comet Way	JAK for automated services	Commercial	Comet Way Ltd.
DECAF	Language or environment for MAS development	Academic	University of Delaware
Grasshopper	Language or environment for MAS development	Commercial	IKV++ Technologies AG
JAFMAS / JIVE	Language or environment for MAS development	Academic	University of Cincinnati
IDOL	Language or environment for MAS development	Commercial	IDOL
IMPACT	Methodology for design/development	Academic	University of Maryland
ZEUS Agent Building Toolkit	Language or environment for MAS development	Commercial	BTexact Technologies
JATLiteBean	Support software	Academic	University of Otago

JESS	Language or environment for MAS development	Academic	Sandia National Laboratories
LEE	MAS modelling/ simulation toolkit	Academic	University of California
Living Markets	Language or environment for MAS development	Commercial	Living Systems AG
MAML	MAS-based modelling/simulation tool	Commercial	Agent-Lab Ltd.
MAP / CSM	Simulation toolkit	Academic	University of Bonn
NARVAL	Framework to develop personal assistants	Commercial	Logilab
RETSINA Agent Foundation Classes (AFC)	Language or environment for MAS development	Academic	Carnegie Mellon University
StarLogo	Language or environment for MAS development	Academic	MIT MEDIA lab
Cybele / Cybele PRO	Support software / Development Framework	Commercial	Intelligent Automation, Inc
AgentTool	MAS engineering approach	Academic	Kansas State University
ASDK		Commercial	IBM's Tokyo Research Laboratory

Table D2.1: Agent related tools and technologies

These toolkits and technologies are quite diverse ranging from specific, proprietary toolkit to open source and freely available development platform from both commercial and academia. Most of the toolkits are mainly for MAS related development and quite a few follow FIPA specifications. It depends on the researcher/developer to choose the right toolkit with suitable technology and programming language. JAVA as a language and platform seems to be the dominant among other languages due to its platform independence, Object Serialization, Remote Method Invocation (RMI), Aglets, threading, sockets, applets (web browser support) and built-in security features. Although few bespoke and smaller scale technologies in .NET, Python and C are also becoming available.

Some of the prominent technologies, explored as part of the research are following:

Java Agent Development Environment (JADE)

JADE is a Java based development platform aimed at developing Multi-Agent Systems. It is fully compliant with FIPA specification (FIPA ACL) for inter-agent communication. The platform enables the development of BDI agents, and provides graphical tools for testing and debugging the developed agents. The agent configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. The JADE communication architecture offers flexible and efficient messaging, by creating and managing a queue of incoming ACL messages, private to each agent; agents can access their queue via a combination of several modes: blocking, polling, timeout and pattern matching based (Bellifemine et al., 2007).

Agents are implemented as one thread per agent, but agents often need to execute parallel tasks. Further to the multi-thread solution, JADE also supports scheduling of cooperative behaviours, where JADE schedules these tasks in a light and effective way. The run-time also includes some ready to use behaviours for the most common tasks in agent programming, such as FIPA interaction protocols, waking under a certain condition, and structuring complex tasks as aggregations of simpler ones. JADE provides a GUI for the remote management, monitoring and controlling of the

status of agents e.g. stop and restart agents. The GUI also allows creating and starting the execution of an agent on a remote host, provided that an agent container is already running (<http://jade.tilab.com>).

There are different types of agent which can be created in JADE;

Dummy Agent: This type of agent is used for inspecting message exchanges among agents. It facilitates validation of an agent interface before integration into the MAS, and also facilitates interrogative testing in the event where an agent is failing. The graphical user interface provides support for editing, composing and communication i.e. send and view ACL messages to/from agents.

Sniffer Agent: This type of agent allows the tracking of agent messages exchanged in a JADE platform. It can sniff an individual agent or a group of agents, and every message directed to or coming from that agent or group of, is tracked and displayed in the sniffer window. The tracked message(s) can be viewed, saved and loaded by user at any time for analysis.

Introspector Agent: It allows the monitoring and control of the life-cycle of a running agent and its exchanged messages including the queue of sent and received messages.

NetLogo

NetLogo is also quite popular agent toolkit/development environment especially as a learning tool. It includes a multi-agent programming language and integrated modelling environment with simulation capabilities to enable exploration of emergent phenomena. It has an extensive library of sample models for variety of domains such as economics, biology, physics, chemistry, psychology, and other natural and social sciences. NetLogo is based on earlier toolkit StarLogo and particularly well suited for modelling complex systems developing over time. Modellers can give instructions to hundreds or thousands of independent agents all operating concurrently. This makes it possible to explore the connection between the micro-level behaviour of individuals and the macro-level patterns that emerge from the interaction of many individuals (Vrobel et al, 2008).

Some of the main features of NetLogo include:

- Cross-platform (Mac, Windows and Linux)
- Run as a standalone application as well as Models can be run as Java applets inside a web browser
- Fully programmable
- Simple language structure
- Language is Logo dialect extended to support agents
- Mobile agents (turtles) move over a grid of stationary agents (patches)
- Create links between turtles to make aggregates, networks, and graphs
- Large vocabulary of built-in language primitives
- Double precision floating point math (IEEE 754)

Recursive Porous Agent Simulation Toolkit (RePast)

RePast is a free, open-source, cross-platform, agent-based modelling and simulation toolkit. One of the great features of the Repast is its implementations in several languages such as Java, Python and .NET, which allows the developers to choose their preferred language. It also has built-in adaptive features such as regression and genetic algorithms.

Agent Based Learning Environment (ABLE)

ABLE was developed by the Research department of IBM which provides functionality similar to JADE but utilises the JavaBeans technology instead. It also uses its own rule based language and provides a set of ready to use artificial intelligence algorithms for quicker development.

VOYAGER

It is a java based agent software support package which is standard neutral. Its core strength is its capabilities to integrate and support other distributed technologies such as CORBA and Web Services. It is based on Object Request Broker (ORB) and offers an extension to Remote Method Invocation (RMI).

Java Agent Template Lite (JATLite)

JATLite is a Java-based package of programs to facilitate the development of agents that exchange messages over the network/internet. It provides Agent Router functionality using its core architecture feature Agent Messaging Router (AMR), which allows any registered agent to send messages to any other registered agent by making a single socket connection to the Agent Router. Messages are therefore seamlessly transferred from the sender to the receiving agent without each knowing the address (or socket connection). It provides a robust message passing system through its AMR and all messages are buffered to avoid data loss in the event of intermittent network problems.

Aglets Software Development Kit (ASDK)

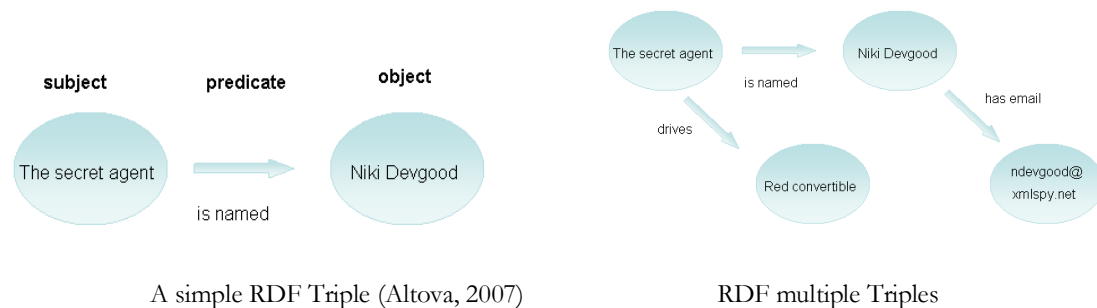
ASDK is a Java based framework and environment for developing and running mobile agents. ASDK can be used for Multi-agent system development; however, it is predominately used for mobile agents in network related projects. It uses Aglet, which is a Java object that can move from one host on a network to another. When the aglet moves it takes the program code with it, as well as all the objects it is carrying.

Appendix E: Ontology Languages Overview

Ontologies are explicitly specified in a formal language and following are some of the widely used languages used.

Resource Description Framework (RDF)

RDF is an official W3C standard for describing resources that exist on the Web, intranets, and extranets. RDF builds on existing XML and Uniform Resource Identifier (URI) technologies, using a URI to identify every resource and make statements about resources. RDF statements describe a resource (identified by a URI), the resource's properties, and the values of those properties. RDF statements are often referred to as “triples” that consist of a subject, predicate, and object, which correspond to a resource (subject) a property (predicate), and a property value (object). (Altova, 2007)



A simple RDF Triple (Altova, 2007)

RDF multiple Triples

The triples with subjects, predicates, and objects, RDF allows machines to make logical assertions based on the associations between subjects and objects. Since RDF uses URIs to identify resources, each resource is tied to a unique definition available on the Web. RDF triples can be written with XML tags. RDF provides only a model and syntax to describe resources; however it does not specify the semantics of the resources, so to define semantics we need RDFS and OWL (Altova, 2007).

RDF Schema (RDFS)

RDF Schema is based on RDF and allows the definition of basic ontology elements such as classes and their hierarchy, properties with their domain, range and hierarchy (McBride, 2004). It is an extensible knowledge representation language for creating vocabularies that describe groups of related RDF resources and the relationships between those resources. It provides basic elements for the description of ontologies or vocabularies intended for structuring RDF resources and the allowable properties within a given domain (Altova, 2007). Based on the RDF triples, RDFS triples consist of classes, class properties, and values that define the classes and relationships between the resources within a particular domain. It also allows classes of resources that share common properties. Resources are defined as instances of classes. A class is a resource too, and any class can be a subclass of another. This hierarchical semantic information is what allows machines to determine the meanings of resources based on their properties and classes. RDFS is well suited for expressing lightweight ontologies (Sabou, 2006).

Web Ontology Language (OWL)

OWL is one of the widely used and richest standard ontology description language (Cardosa, 2007) as well as W3C specification for creating Semantic Web applications. It is built upon RDF and RDFS, and defines the types of relationships that can be expressed in RDF using an XML vocabulary to indicate the hierarchies and relationships between different resources. OWL is a more expressive ontology language than RDFS. The basis for developing OWL was the DAML+OIL language which originated by merging two language proposals that aimed at overcoming the expressivity limitations of RDF(S): DAML-ONT and OIL (Horrocks et al., 2003). OWL enhances the expressivity of RDF(S) providing means to describe relations between classes (e.g., disjoints, union, intersection), cardinality and value restrictions on properties (e.g., cardinality, universal and existential quantifiers), property characteristics (e.g., transitivity, symmetry), equality etc.

Semantic Web ontologies consist of taxonomy (system of classification) and a set of inference rules from which machines can make logical conclusions. Since taxonomies express the hierarchical relationships that exist between resources, we can use OWL to assign properties to classes of resources and allow their subclasses to inherit the same properties. OWL also utilises the XML Schema data types and supports class axioms such as `subClassOf`, `disjointWith`, etc., and class descriptions such as `unionOf`, `intersectionOf`, etc.

Appendix F: Relevant Tools & Technologies Review

ITS based systems vary from a simple on-off sensor to highly complex central command server systems, and may use range of communication methods, technologies and standards. Obviously, the tools & technologies for the design and development of ITS systems are also diverse. Each technology has its advantages and disadvantages in terms of its capabilities, approach, target platform and availability. The following is the review of some of the technologies considered or used in ITS@CU platform development.

Programming Languages and Technologies

Java: It is an object-oriented language by Sun Microsystems also derived from C++. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. Java also supports Multi-Agent programming in form of JADE and Aglets. In most research level projects Java has been mostly the first choice as a programming language including for agent based systems.

C++: It is an object-oriented language originally derived from C. It is quite powerful language however it is rather difficult to program as compared to C# and Java. It is still used in various development platforms such as .NET compact framework and devices and hardware programming.

C#: It is a very powerful, object-oriented, and type safe programming language, derived from C/C++. It is the main language in the popular .NET framework by Microsoft, and combines the raw power of C++ with the high productivity of Visual Basic.

The author decided to use C# for the following reasons:

- Single language to support all Microsoft based platforms (mobile, desktop and web) and using single IDE (Visual Studio 2008) for the development which is important to avoid learning and implementation of multiple languages and development tools.
- It is widely used in industry for high level commercial projects and proven highly successful.
- C# supports completely the entire object oriented paradigm and integrates other powerful features.
- C# also support component-oriented and Service-oriented development
- Good capability to implement Agent oriented approach
- Powerful library .NET Framework 2.0/3.5 and tools
- Excellent help and support through Microsoft MSDN website
- T@lecom's other systems are also implemented in C#, which means the integration of the project with company other system will be easier.

Extensible Mark-up Language (XML): XML was designed for structuring hierarchical data in the file in a way that will be self-descriptive and easy to process by computers (XML Working Group 2006). The XML document is structured in a tree of elements called tags (e.g. <book>). Each element can contain data, other tags, comments, character references and processing instructions (XML Working Group 2006). Each document contains also a declaration that defines which XML version of the standard (W3C defines 1.0 and 1.1) it follows.

There are two types of correctness of XML document. Well-formed document conforms to the XML syntax rules. The document is Valid when it is semantically correct. The semantical structure of the document can be defined using Document Type Definition document or XML Schema document. The first technology has been used for a long time and also the FIPA ACL XML Message Specification uses it (FIPA 2002d) but at the same time it provides only a loose definition of the XML structure and requires a separate parser (it's not formed using XML structure). The XML Schema is a newer technology that can be parsed with XML parsers, because it is formed as XML document. It also provides more strict way to define the XML structure with use of restrictions and data modifiers which is a great advantage but on the other hand because it has fewer tools than the DTD.

Development Frameworks

There are numerous development frameworks which can be utilised in the development of ITS based systems. Following are some of the frameworks:

Java Development Environment: Java is a platform independent i.e. it is not specific to any one processor or operating system, but rather an execution engine called a Java Virtual Machine (JVM) and a compiler with a set of standard libraries implemented for various hardware and operating systems so that Java programs can run identically on different OS.

The Java platform consists of several programs or components each of which provides different capabilities. For example, the Java compiler converts Java source code into Java bytecode (an intermediate language for JVM), is provided as part of the Java Development Kit (JDK). The Java Runtime Environment (JRE), complementing the JVM with a just-in-time (JIT) compiler, converts intermediate bytecode into native machine code on the fly. The JDK also provide extensive libraries allowing developers with pre-compiled code and faster development.

There are different editions targeting different type of the development areas or platforms.

- Java SE (Standard Edition): Intended for general purpose use on desktop PCs, servers and similar devices.
- Java EE (Enterprise Edition): Java SE plus various APIs useful for multi-tier client-server enterprise applications.
- Java ME (Micro Edition): Intended for embedded/mobile devices based application development. It has limited set of libraries compared to SE or EE editions due to smaller memory and storage limitation of the mobile devices.
- Java Card: Allows small Java-based applications (applets) to be run securely on smart cards and similar small memory footprint devices.

.NET Framework: .NET is a core framework for all windows based development by Microsoft. It is very comprehensive development framework comprising of development tools, run-time environments, server infrastructure, powerful Integrated Development Environment (IDE) support and intelligent tools, which enable the application development for various platforms and devices. It supports various languages such as C#, C++ and Visual Basic.

The current version of the .NET Framework is 3.5 which include new set of features such as Windows Workflow Foundation (WF), Windows Communication Foundation (WCF), Windows Presentation Foundation (WPF) and Windows CardSpace. .NET supports the SOA infrastructure and provides powerful services integration support in form of XML Web Services, Remoting, COM and WCF.

The .NET Framework consists of two main components: the .NET Framework class library and the Common Language Runtime (CLR). The .NET Framework class library provides the types that are common to all .NET languages. Programmers can use these types to develop different kinds of applications, such as console applications, Windows and Web Forms, and XML Web services. The CLR consists of components that load the code of a program into the runtime, compile the code into native code, execute and manage the code, enforce security and type safety, and provide thread support and other useful services.

In this research, .NET has been predominantly used for the development of various systems and its component. It is highly established technology and supported by comprehensive class library and development support in form of Microsoft Developer Network (MSDN) and top class development tools such as Microsoft Visual Studio 2008/10. It is equally suited for all flavours of development platforms i.e. desktop, web and mobile. It also provides inherent security in form of code access security and also supporting all major security protocols. It is highly interoperable, extendible, portable and with excellent memory management. New addition to the framework such as Language integrated Queries (LINQ), AJAX, Windows Communication Foundation (WCF), parallel and multi-core computing support makes it even more appealing. In SOA environment WCF plays very key role and allows .NET based application to integrate easily with virtually any other system. It is also crucial for the development of cloud computing based application for the Microsoft's Windows Azure platform.

Other major reason for adopting .Net is to complement the existing T@lecom systems which are also implemented in .NET which means easier integration of the project with company's other systems.

Mobile Development Frameworks

The ITS systems also includes range in-vehicle and infrastructure based devices either standalone or embedded. It performs variety of tasks from simple on/off sensor to complex task and utilise various communication channels such as wireless (Wi-Fi, infrared, Bluetooth etc.). Mobile or embedded devices as compared to other systems have limited memory, processing speed, power and storage, which mean applications running on such devices must have smaller footprint. Therefore, the development frameworks for mobile platforms are generally different to address the limitations.

There are different development frameworks from different vendors for mobile or compact devices development. Some of the major frameworks investigated in this research include:

Java ME (Micro Edition): It is a Java based framework for the development of small footprint and mobile applications. It provides a robust, flexible environment for applications running on mobile and other embedded devices such as mobile phones, personal digital assistants (PDAs), TV set-top boxes, and printers. Java ME include flexible user interfaces, security, support for network protocols and networked and offline applications. Java ME applications are highly portable across many devices due to Java's inherent platform independent feature.

Java ME includes J2ME toolkit which is a state-of-the-art toolbox for developing mobile applications. It integrates CLDC, CDC and Blu-ray Disc Java (BD-J) technology into one SDK. Java ME SDK 3.0 is the successor to the popular Java Wireless Toolkit 2.5.2 and Java Toolkit 1.0 for CDC. It provides device emulation, a standalone development environment and a set of utilities for rapid development of Java ME applications. Recent version also provides good support for 2D/3D graphics, imaging, multiple I/O capabilities and range of wireless communication methods and channels.

It is also supported by industry leading IDE's such as Eclipse, NetBeans (Mobility Pack) and JRocket, and also provide extensive class libraries and packages which allows easier development.

Android: It is relatively new platform however gaining a lot ground in mobile and embedded arena. It is Linux-based platform with application layer programming exclusively done in Java. It provides extensive Java SDK for the development and supports Eclipse and NetBeans IDE. Android is targeted for Smartphone development however it can also be used other communication based application on mobile platform and appliances.

iPhone: Platform for Apple based devices such as iPhone, iTouch and iPad and uses Objective C which is based on the C programming language. It is vendor specific and closely controlled by Apple.

BlackBerry JDE: It is a Java based platform for Blackberry devices by Research In Motion (RIM). It supports most of the J2ME application however applications must be blackberry supported i.e. conforms to its UI and standards. Security and wireless data reliability is the key features of the Blackberry JDE. However, the platform is focused on the Smartphone devices only.

.NET Compact Framework: It is a cut down version of .NET platform specifically designed for Microsoft Windows CE based mobile/embedded devices such as PDAs, mobile phones, factory controllers, set-top boxes, etc. It supports all the Microsoft tools and languages such as C#, VB.NET and ability to use Web Services and IDE Visual Studio 2008. It uses some of the same class libraries as the full .NET Framework and also a few libraries designed specifically for mobile devices such as Windows CE. It is the most dominant framework for enterprise mobility solutions.

In this research, .NET Compact Framework has been predominantly used for the development of mobile applications. It is highly established technology and supported by comprehensive class library and development support in form of Microsoft Developer Network (MSDN) and top class development tools such as Microsoft Visual Studio 2008. It also provides inherent security in form of code access security and also supports all major security protocols. It is highly interoperable, extendible, portable and with excellent memory management. New addition to the framework such as Language integrated Queries (LINQ) and Windows Communication Foundation (WCF) features make it even more appealing. In SOA environment WCF plays very key role and allows .NET based application to integrate easily with virtually any other system. It is also crucial for the development of cloud computing based application for the Microsoft's Windows Azure platform.

Other major reason for adopting .NET is to complement the existing T@lecom systems which are also implemented in .NET which means easier integration of the project with company's other systems.

Integrated Development Environment (IDE)

Eclipse: An open source Java IDE allowing the development of enterprise, embedded/Device, Rich Client and Internet applications. It supports the complete Application Lifecycle Management (ALM) and Service Oriented Architecture (SOA). It has a large open source community and supported by a large ecosystem of commercial developers, universities and research institutions.

NetBeans: It is a free, open-source IDE for developing desktop, enterprise, web, and mobile applications with the Java language, C/C++, and even dynamic languages such as PHP, JavaScript, Groovy, and Ruby and supports all major platforms including Windows, Linux, Mac OS X and Solaris.

Microsoft Visual Studio: Visual studio is a very powerful and most widely used commercial IDE by Microsoft. It allows the development of standalone applications (desktop/server), mobile applications, embedded devices applications, web applications and web services predominately for Microsoft .NET platform. The current version is Visual Studio 2010 which provides advanced development tools, debugging features, database functionality, smart client connected applications, visual designing and testing features.

In this research, mostly Visual studio 2005 and 2008 version was used due to its native support for .Net framework and also XML web services for SOA implementation. It was also a requirement by T@lecom as they use Visual Studio for most of their development.

Ontology editors

There are several ontology editors available such as Protégé, Swoop, OntoEdit and Altova SemanticWorks. However Protégé is the most widely used editor and currently its share is about 62.8% (Cardosa, 2007).

Protégé is a Java based free, open source ontology editor and knowledge-base framework. Protégé is supported by a strong community of developers and academic, government and corporate users, who are using Protégé for knowledge solutions in areas as diverse as biomedicine, intelligence gathering, and corporate modelling. The Protégé platform supports two main ways of modelling ontologies via the Protégé-Frames and Protégé-OWL editors. Protégé is also used in this project due to the following reasons:

- Extensible and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development.
- Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema.
- Excellent documentation and examples for quick start.

Database Systems

ITS system or subsystems whether a small device or large server systems, requires some form of data storage and intelligent data manipulation in order to function. Databases are therefore backbone for most systems and efficient database design and implementation is vital to the success of any system. There are many data storage and manipulation technologies from simple flat file to highly complex database management systems.

Following are some of the database systems evaluated for the research:

Microsoft SQL Server 2005/2008: Microsoft SQL Server is one of the widely used relational database management system (RDBMS). It is primarily used for Microsoft based development platform however different other platform can utilise it equally well. The primary query language used is Transact-SQL which is based in SQL with additional built-in functions and programming features. SQL Server is optimised for the multithreaded, pre-emptive multiprocessing kernel of Windows NT, providing concurrent access to high volumes of data. The most recent version is SQL Server 2008.

Some of the highlights of the SQL Server are:

- Native support for .NET platform and Visual Studio
- Highly portable, reliable and flexible
- Client/server orientation
- Set of UI tools for database design, management, analysis, tuning, performance evaluation
- Ability to handle very large databases (VLDB)
- Good for Knowledge Base and Data Mining
- Robust feature for Integration and multi system synchronisation

- Resilient and automatic recovery and backup features
- Reporting, Business Intelligence and data Analysis
- Spatial data support for GIS and location aware systems
- Mobile Version also available (SQL Compact Edition)

Oracle RDBMS: It is a Relational Database Management System (RDBMS) from Oracle. It is market leader in enterprise and data warehousing sector (source: RDBMS, Gartner Report). It runs on variety of platforms from micro to mainframe. The current version is 11g and it includes built-in change testing, capability of viewing tables back in time, superior compression of all types of data and enhanced disaster recovery functions. It provides built-in Java Virtual Machine (JVM) support. It is also highly efficient in grid and cluster environment making it ideal for distributed systems. The query language is PL/SQL however Java can be used for complex functions due to its native Java support.

MySQL: MySQL is also a Relational Database Management System (RDBMS) now acquired by Oracle. It is relatively light weight, open source and free to use. It is quite popular among web development community due to its simplicity and free licensing terms. It is gradually growing into enterprise sector as well. It is supported on all major platforms and provides variety of tools for managing and implementing database.

SQL Compact Edition: SQL Server Compact Edition is a smaller footprint version of SQL Server and it is specially designed for compact devices and systems with limited processing power. It is relational database system and supported by all the .NET based application with security, reliability, efficiency, manageability of the SQL Server but with much smaller file size which makes it ideal for the embedded and mobile devices. Some of the highlights of SQL Server Compact Edition are:

- Great mobility features such as Synchronisation in form of Merge Replication and Remote Data Access (RDA)
- Portable be-spoke and enterprise applications
- Ideal for occasionally-connected & offline use especially in wireless communication where connection is never reliable
- Can be embedded in applications & devices as a whole database stores itself as a file

In this research SQL Server Compact Edition (version 3.5) has been used as data storage for all mobile applications. Beside the key benefits mentioned above, it provided simple yet powerful integration with Control Systems and easier communication with other systems in form of data synchronisation methods. Its native support for .NET and Visual studio and also familiar implementation of the full SQL Server were also the key factors for adopting it as main database storage for mobile devices.

Appendix G: Wireless Communication Technologies

1. Wireless Technologies Analysis

Wireless technologies are revolutionising the process of traffic data collection and transmission between various ITS components i.e. Vehicle, road Infrastructure and central control system. Major data bearer and wireless communication technologies used in communication between ITS components includes GPRS, 3G, HSPA, Wi-Fi, Bluetooth, IrDA and TETRA. Some of the successful implementation includes:

- Fleet Management and communication using GPRS/3G/HSPA based on-board vehicle devices/mobile applications (WD Live Fleet).
- Radio Data Service-Traffic Messaging Channel (RDS-TMC) for traffic information announcements (iTIS, TrafficMaster).
- Real-Time vehicle tracking and satellite navigation systems (MasterNaut, WD Live Tracker).

In this research, various wireless technologies were investigated such as GPRS/3G for cellular mobile data communication, Wi-Fi for localised high speed data transfer, and Bluetooth and IrDA for short range Personal Area Network (PAN), and TETRA for highly secure, reliable, long range transmission. Each technology has different data rate/speed, range/coverage and reliability characteristics and used for different purposes.

Name	Frequency [MHz]	Download /Upload [kb/s]	Range [m]	Latency [ms]	Availability / Coverage	Advantages	Disadvantage
TETRA	380–390	7.2 / 7.2	8000	500	90+%	Coverage, Security, Reliability	Cost, Need to allocate one channel
Satellite	137–150	4.8 / 2.4	huge	huge	100% on flat areas	Coverage	Available only in open areas, Cost
FM DAB,	87–110	10 / —	15	—	99% of population + motorways	High speed vehicle support	One way communication
T-DMB	209–230	192 / —	000				
DVB- H	209–230	72 / —			90% of population + motorways		
DRM	Below 30						
GSM Voice	890–915,	9.6	1250	5000	Very good coverage in urban areas	High speed vehicle support	Provider dependent
SMS	935–960,	160char/~2.5s		700			
HSCSD		112				In-expensive	
GPRS		160					
EDGE	1710 – 1785	320	580	600-300	Available mainly in urban areas	data rate	

W- CDMA	2000/3500	350	380	200	Available	data rate	
HSPA	2000	>2048 / 350	550	150	Introduced	data rate	
802.11a b, g 802.11p	, 2412 – 2472 5850 – 5925	23000, 4500,19000	90		Popular for home use Standardised	In-expensive	Access “on the pause”
Wi- Max	2300	24 000 / 4096	400		Tested	Data rate and travel speed	Not fully implemented
FLASH OFDM iBurst	- 450 1800	5300 64000	3000		Not available		Availability

Table 1: Comparison of wireless communication

In the context of vehicles which is a moving object requires wireless technologies with longer range however intra-vehicle communication between devices can be achieved by small range technologies such as Bluetooth.

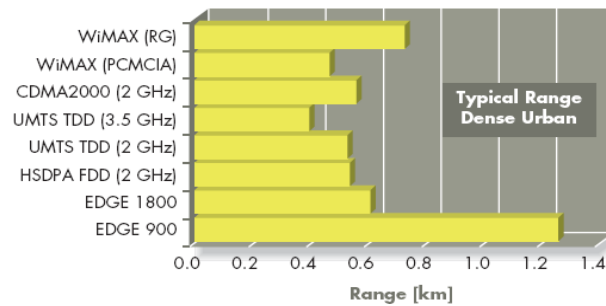


Figure 1: Typical range (uplink data rate 64 kb/s) (Hurel et al.,2005)

WiMax and LTE are quite interesting new technologies with a potential to provide wide range and high speed bandwidth. These technologies can be utilised in ITS components for reliable communication. However, in this research the implementation of the systems will be independent of the type/data bearer, so only the available technologies 3G/GPRS, TETRA, Wi-Fi and Bluetooth are used.

2. Vehicle-Vehicle-Infrastructure Communication in ITS

Vehicles and roadside systems are usually autonomous however one of the main idea behind ITS is to move towards a connected network of vehicles and infrastructure systems where communication takes place between vehicles and infrastructure. ITS based systems by nature are highly distributed, geographically dispersed and multi layered/components, therefore efficient use of communication technologies are vital at every level of systems integration, traffic data collection, and communication between components/devices. *In ITS, the communication is usually categorised in following three ways:*

- Vehicle-to-Infrastructure (V2I)
- Vehicle-to-Vehicle (V2V)
- Intra-Vehicle

V2V and V2I communications can provide benefits in terms of safety, allowing constant, real-time sharing of information that will help warn of hazards (Active traffic management) and also potentially maintain the flow of traffic (Intelligent routing and signal adaptation). In this way the systems will provide a new dimension in inter-system communication and distributed control systems. It will also provide vehicles with effective internet connectivity through roadside gateways and grids.

Ad-hoc V2V and V2I networks

V2V communication allows vehicles to interact with each other wirelessly and sharing information. In the context of V2V, one interesting idea is the formation of ad-hoc and temporary network between vehicles (VANET) in communicating range. These ad-hoc networks can be used for self-organising and various other such tasks, where each vehicle and infrastructure controls act as a network node (on the ad-hoc network) and perform various actions however, within the constraint of the infrastructure to ensure safety and security. The communication messages between vehicles/nodes can be direct or can propagates to destination using a number of intermediate links. If vehicle mobility causes links to break, message can be re-routed using a different path.

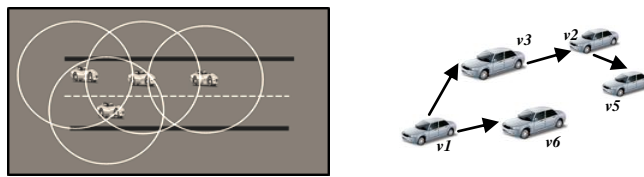


Figure 2: Vehicle based ad-hoc networks

The major benefit of such ad-hoc network is the decreased dependency on fixed infrastructure in order to create highly connected ITS environment. In this way local services can also be provided to the nodes i.e. vehicles. Major challenges in achieving the ad-hoc network include:

- Network congestion
- Security
- Range allocation and efficiency
- Level of fixed infrastructure required
- Application dependent
- Routing protocol

The new generation of Bluetooth v3 especially v4 provides a potential for achieving such ad-hoc networks by forming Personal Area Networks (PAN) of nodes within range and with improved reliability. These ad-hoc PANs can join together by sharing bridging nodes to form large even larger networks.

In this research, the idea of ad-hoc networks in the context of ITS was investigated, and various Vehicle to Vehicle (V2V) technologies were analysed as part of the ITS@CU platform development. As a result an experimental mobile application was developed for simulating Vehicle Controls forming ad-hoc Personal Area Networks (PAN) using Bluetooth. The application was developed using the Mobile Application Development Framework (described in *chapter 7, section 7.4*) and it was predominately used for testing the communications between vehicle-to-vehicle and vehicle-to-traffic-controls. The Ad-hoc V2V idea was taken to further level by incorporating Agents to interact between participating nodes and make decisions by cooperating with each other.

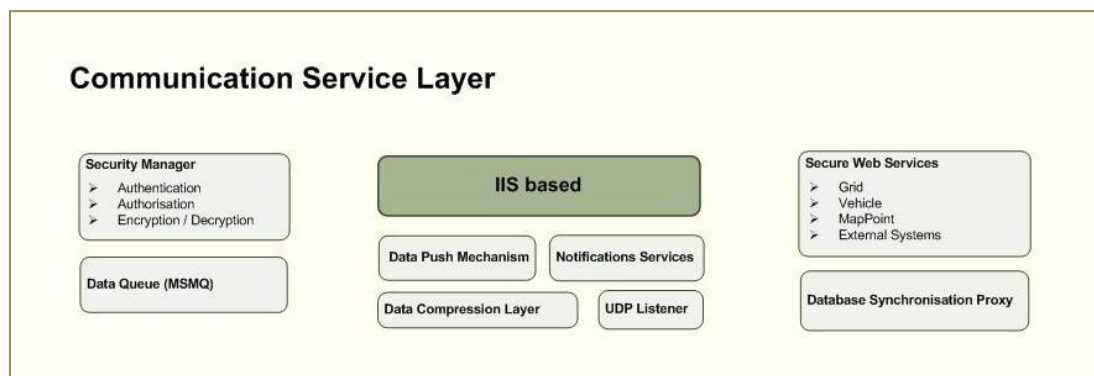
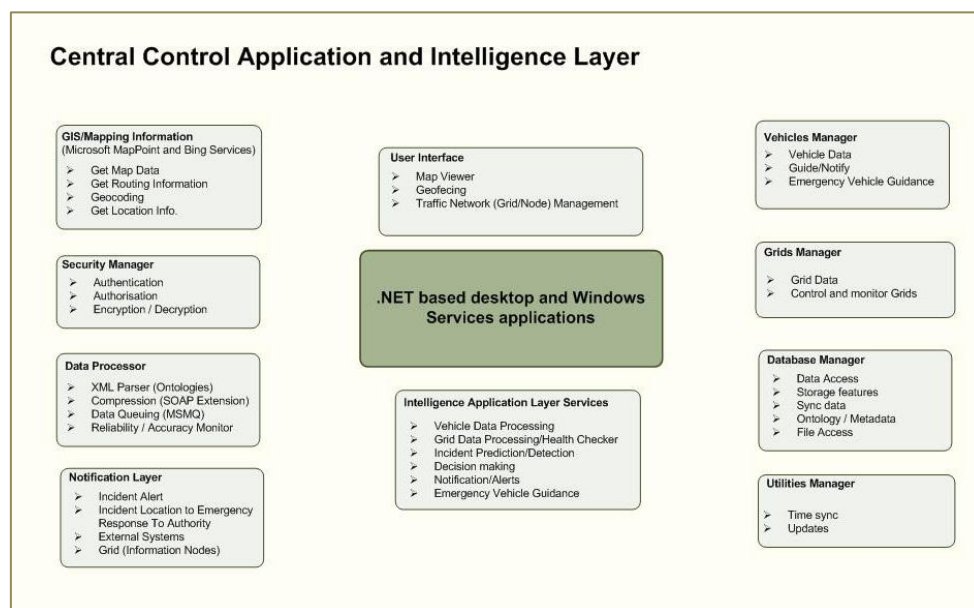
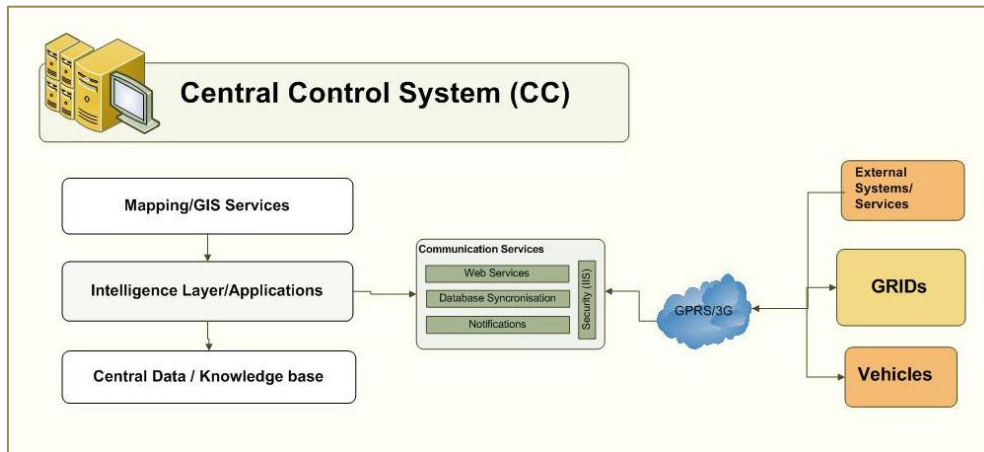


Figure 3: Ad-hoc V2V PAN Application

Left: Application initial setup; *Centre:* Bluetooth setup and ad-hoc vehicle/device in range info;
Right: Customised V2V Messages for simulation/Test purposes

Appendix H: ITS@CU Components design details

1. Central Control System



Data and Knowledge base

Vehicles Information

- Identification, type, size/measurements
- Mileage, MOT, Tax, Registration etc.
- Location information

Ontology Information

- XML based metadata files for specifying common ontology

Infrastructure/Environmental Data

- Traffic network Information
- Weather data (Historical)

GRIDs

- Grid ID, boundary / Geo-fencing
- Grid Nodes, Components, Segments etc.

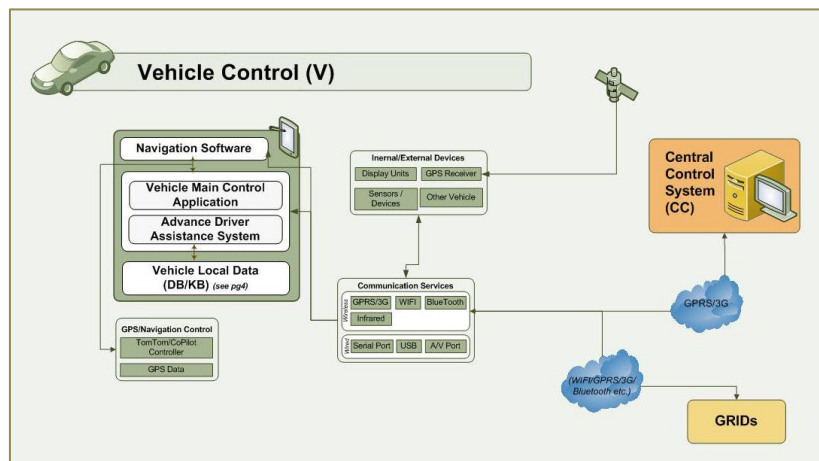
Application Data

- Application specific data

GIS/Mapping & traffic Data

- Geo-codes
- Map data

2. Vehicle Control System



Vehicle Main Control Application

GPS/Navigation Controller

- Navigation Software Controller (TomTom/CoPilot)
- GPS data layer (NMEA GPS data/Fix converter and bundler)

Security Manager

- Authenticate
- Authorising
- Encryption / Decryption

Data Processor

- XML Parser
- Compression (SOAP Extension)
- Data Queuing
- Reliability / Accuracy Monitor

Mobile Application Platform For Windows CE/Mobile 5/6 based on .NET Compact Framework

Intelligence Application Layer

- Intra-Vehicle data management
- Prediction algorithm
- Decision making algorithms
- Notification Layer

Utilities Manager

- Time sync
- Update Services

Connection Manager

- Wireless Network/Connection establishment, Monitoring, prioritizing etc.
- Intra-Vehicle, Vehicle to Vehicle, GRID and CC

Data Manager

- Data Access (SQL CE)
- Replication/Synchronisation with Server (CC)
- Knowledge Base / Ontology
- File Manipulation

Communication Manager

- Web Services Layer
- TCP Layer
- UDP Layer
- SQL CE Replication/ RDA Layer

Advance Driver Assistance System

Smart Navigation Control

- Advance congestion/road block warning with pre-configured re-routing
- Pre-Configured / One touch navigation
- Customised local information

Display / UI

- Messaging/Alerts/Info display (from intra-Vehicle, V2V, V2i, V2C)
- Interactive response
- Status update (to CC)

Guidance system

- Advance guidance system

Vehicle data

Vehicle Information

- Identification, type, size/measurements
- Mileage, MOT, Tax, Registration etc.
- Intra-Vehicle devices info
- Current Status of Intra-Vehicle devices

Vehicle Behaviour

- Vehicle drive times, mileage Averages,
- Accidents/Breakdowns, Device malfunctions

Driver Behaviour

- Drive times and area
- Averages (Speed, mileage, manoeuvre, Acceleration etc.)
- Driver and Intra-Vehicle systems interaction

Vehicle Location

- Current GPS based location data
- Historical location data

Ontology Information

- XML based metadata files for specifying common ontology

Application Data

- Application specific data

Appendix I: Mobile Application Development Framework (MADF) Description

This section describes the API description of the mobile application framework developed as part of the KTP project and used for simulation of various traffic controllers in ITS@CU platform. The API provides details of all of the elements necessary for any application to be built within the platform.

Overview

The architecture is logically split into Business Logic, Resource and Data and User Interface (UI) layers. Each layer deals with a different element of the platform. Segregation between the different layers will allow for easier modification when necessary at a later stage.

User Interface Layer

The UI layer contains Views and Presenters.

- **Views** are controls on the screen that the user can interact with.
- **Presenters** are classes that drive the views

Business Logic Layer

- **Business Entities** are classes that represent business concepts, such as “Customer”, “Bank Account” or “Address”.
- **Business Logic Components** are classes that implement the majority of the business logic within the application.
- **Business Workflows** drive the overall flow of control within the application.

Data Access & Resource Layer

- **Data Access Components** can fetch data from a local store such as SQL CE, and they may use infrastructure components to manage data subscriptions and expirations.
- **Service Agents** act as proxies to external web services and meet the additional challenge of working in occasionally connected environments.

Since the mobile application framework relies heavily on access to an SQL CE database, any direct access to this database would be governed under this layer. There is also the need to access a variety of text or XML files for configuration or logging purposes or otherwise. The Data layer should contain generic handlers for this file access.

Management Layer

The Management layer contains Logging, Configuration and Error management.

Connectivity Layer

Connection and Network Management includes providing features for assessing the current connectivity of the device and reacting to changes in connectivity.

Security Layer

The Connectivity layer contains Authorisation and authentication features. It also provides encryption using AES 128 encryption.

Components

There are various components to the application that are needed. Some of these components should only ever exist once across the application, whilst others should be available in as many instances as necessary. Logical components can have various elements to them that exist across the different layers in the application. These would then be broken down into separate smaller components that deal with the specific elements relating to a single layer.

A component can either be:

- Static – a single instance, but accessible from across the whole application.
- Hybrid – many instances that access some instance-specific and some global properties.
- Standard – many instances of this class can exist and each instance is stand-alone.

Merge Replication

The Merge Replication class allows the application to synchronise data with the server using merge replication. Since this may happen at any point within the application, this component should be global.

There are two types of merge replication that can occur. The first is a “reset” replication, where the local database is removed and recreated based on the replication with the server. The second is a “standard” replication where the local database is synchronised with the server one. In the case of a “standard” replication, the replication is either two-way or upload only.

If a user has logged out then the synchronisation will need to create a new subscription.

The user interface may need to indirectly trigger a merge replication, but it is far more likely that an event within the business layer will trigger a replication.

When a replication event starts, the Device Status will need to be updated to show that a replication is in progress. Once the replication is completed, the Device Status will need to be updated to show that a replication is now complete.

The Merge Replication class is predominantly a Business layer class. It will need to interact with the merge replication functionality that is contained in the Data layer to perform the actual merge replication process.

Public Methods:

- `Reset()` – This will trigger a reset of the database and then recreate it from the server.
- `Synchronise(ExchangeType Type, bool InitialiseSubscription)` – Synchronise the database or upload, depending on the flag.

Public Events:

- `Started()` – Event occurs when the synchronisation process has started.
- `Complete()` – Event occurs when the synchronisation process has stopped.
- `Error()` – Event occurs when an error occurs during synchronisation.

Public Property:

- `InProgress {get}` – Whether a synchronisation attempt is in progress.

Device

The Device class allows the application to readily access various device methods and properties. Since there is only one device, this class should be global. The class should make available the Device ID (whether this is the IMEI or other), the GPS status, the Connectivity status, the Offline status and the Synchronisation status. It should also make available the sounds, clock and phone functionality.

The class does not contain any business logic and is therefore in the Data layer.

Public Methods:

- Dial(string phonenumber, bool prompt) – dial the provided phone number.
- SMSSendMessage(string number, string message) – Send an SMS message.
- PlaySound(string filepath) – play the sound from the file provided.
- PlaySound(SoundType beep|alarm|notify) – play a standard sound of the selected type.

Public Properties:

- DeviceID {get}
- DeviceName {get}
- IMEI {get}
- SIMNumber {get}
- Memory {get} – Return memory structure (Available and Total).
- OSVersion {get}
- CFVersion {get}
- DateTime {get,set}
- BatteryLevel {get}
- BatteryStatus {get}

SNAPI

This static class monitors the snapi and provides updates to other classes.

Public Events:

- PhoneGPRSCoverage_Changed()
- ConnectionsCellularCount_Changed()
- ConnectionsNetworkCount_Changed()
- PowerBatteryState_Changed()
- PowerBatteryStrength_Changed()

Public Properties:

- PhoneGPRSCoverage {get}
- ConnectionsCellularCount {get}
- ConnectionsNetworkCount {get}
- PowerBatteryState {get}
- PowerBatteryStrength {get}

Device Status Bar

The device status bar is a UI element which updates based on the Device settings. It is found in the UI layer on the device and is a Standard class. It shows the user, in order, the offline status, if the battery level is low, the GPS status, the synchronisation status and the connectivity status of the device.

The UI element interacts with the Global Device class to obtain the settings for the device and keep the display up to date.

The graphics for the various icons in the status bar are embedded into the class, although the background graphic should be readily available. Currently, the control has a minimum size which is the size of the five images embedded in the control.

Public Properties:

- Standard UI control properties (size, location, etc).
- BackgroundImage {get,set} – The image to use on the background of the control.
- BackgroundImageFormat {get,set} – Whether to centre, stretch or tile the image.
- BackgroundColor {get,set} – The colour of the background if no image is selected.
- OfflineIcon {set}
- BatteryLowIcon {set}
- GPSIcon {set}
- SynchronisationIcon {set}
- ConnectionIcon {set}

Connection

The connection class is the only current class in the Connectivity layer. It is a global class and should only be accessible (for now) through the Device class (also global).

The class should allow the user to check the connectivity of the device through checking the GPRS, WiFi, 3G, Fixed line and other potential connectivity options. Currently, many of these options are not implemented.

Public Methods:

- HasConnection() – Check if the device has a viable connection.
- HasCellularConnection() – Check if the device has a viable cellular connection.
- CheckServer() – End to end test of server connectivity.

Ping

This standard Business layer class gathers all necessary information and then sends it to the server using a web service method. The class will almost certainly be triggered using a UI timer on the main form.

Public Methods:

- GeneratePing() – Create a ping and attempt to transmit it.

Error Log

This global Data class enables the logging of error information.

Public Methods:

- Log(string message)

LoginLogoutProcess

This standard Business layer class controls the logging in process. When the application starts, or when a user logs off, this process should be triggered. Only once the logging in process is complete should activity be passed back to the calling class.

In practise, this functionality will be achieved through opening the Login form in Dialog mode so that the form must be closed before focus can be returned to the activating form. Whether this requires a separate class to perform this is debatable.

Public Methods:

- ShowLogin() – Attempt to log in to the application (this will log out of the application if already logged in).
- LoginSuccess() – Event occurs on a successful login.
- Logout() – Event fired to trigger logout related actions.

Public Events:

- LoginSuccess() – Event occurs on a successful login.
- Logout() – Event fired to trigger logout related actions.

Login

This UI class requires the facility to enter a username and a password and pass this information to the relevant (Login) class. On a successful log in, the form should close, otherwise the form should display useful information to the user and remain open, allowing the user to try again.

The form should also allow for exiting the application completely if required and accessing the configuration to allow for changing of IP addresses etc.

The Business layer class requires a username and a password to pass through to the web service. It will respond to the UI component through events to inform as to the success or failure of the login attempt.

Public Methods:

- AttemptLogin(string username, string password)

Public Events:

- LoginSuccess() – Event occurs when the login attempt is successful.
- LoginFailure() – Event occurs when the login attempt fails.

Public Properties:

- LastLoginFailReason {get} – The reason for the last login failure event.

Web Services

The Web Services need to be able to send web service requests to the server. Any response from the web service may need to be dealt with.

Public Methods:

- string WSVersion(bool Synchronous)
- string DeviceLogin(bool Synchronous, string LoginID, string Password, string DeviceID, string AppVersion)
- DeviceStatusStruct DeviceStatus(bool Synchronous, string LoginID, int PushProcessed, string Time, string Battery, string Conn, string Gps, int Lng, int Lat, int Spd, int Dir) – This method will populate the DeviceStatusStruct structure with relevant information or the structure will be made available by the event.
- bool DeviceCommandsUpdate(bool Synchronous, string LoginID, int LogRequired, string Message, string KickOutUser, int WipeData)
- DateTime CurrentDateTime(bool Synchronous) – Get the current date and time from the server.
- string CheckUpgrades(bool Synchronous, string AppVersion)
- FileStruct GetFile(bool Synchronous, string filename) – The FileStruct type contains both the binary data and the hash value for the file.
- bool PutFile(bool Synchronous, byte[] buffer, string filename)
- string UpdateOnSync(bool Synchronous, string UserID, string CompanyDB)

Public Events:

- WSVersionComplete(string result) – Event fires when the asynchronous WSVersion completes.
- DeviceLoginComplete(string result) – Event fires when the asynchronous DeviceLogin completes.
- DeviceStatusComplete(DeviceStatusStruct dst) – Event fires when the asynchronous DeviceStatus completes. Dst is populated with the results.
- DeviceCommandsUpdateComplete(bool result) – Event fires when the asynchronous DeviceCommandsUpdate completes.
- CurrentDateTimeComplete(DateTime dt) – Event fires when the asynchronous CurrentDateTime completes.
- CheckUpgradesComplete(string result) – Event fires when the asynchronous CheckUpgrades completes.
- GetFileComplete(FileStruct fs) – Event fires when the asynchronous GetFile completes.
- PutFileComplete(bool result) – Event fires when the asynchronous PutFile completes.
- UpdateOnSyncComplete(string result) – Event fires when the asynchronous UpdateOnSync completes.

GPS

The global GPS class needs to have the facility to provide co-ordinates for tracking purposes and (potentially) open a satellite navigation program to navigate to a specified location.

Public Methods:

- Start() – Start the GPS driver
- Stop() – Stop the GPS driver
- Navigate(string Label, int Long, int Lat, GPSAddress Address) – Navigate to the provided latitude/longitude or address. Priority is decided by the driver.
- Show() – Show the satellite navigation application.
- Hide() – Hide the satellite navigation application.
- Dispose() – Dispose of the driver.

Public Properties:

- GetCurrentFix {get} – Obtain a current fix from the driver.

Appendix J: CD Contents and projects Source Code

The ITS@CU platform and associated utilities comprises of a large amount of code which is not appropriate to include in the report. A password protected and encrypted CD is therefore enclosed with this report.

It contains the following items in compressed .RAR format (WinRAR software is required):

1. ITS@CU Gateway, Simulation and Mobile Application Development Framework

File: **ITSCU vss code.rar**

Requirements to run/view:

- Microsoft Visual Studio 2008 Professional edition
- SQL Server Compact Edition 3.5 installed and configured
- Windows Mobile 6.1 or higher SDK installed including emulators
- SQL Server 2005/08 Standard/Enterprise Edition with merge replication enabled
- IIS 6 or higher with ASP.NET and SQL replication proxy allowed

NOTE: Commercially sensitive code is not included.

2. Incident Detection System Source Code

File: **Extra Utilities code.rar**

Mobile client application

Requirements to run/view:

- Microsoft Visual Studio 2008 Professional edition
- Windows Mobile 5 or higher SDK installed including emulators
- Microsoft ActiveSync 4.5 Installed (If running windows XP)
- PDA with GPS receiver built-in or attached via Bluetooth
- PDA also needs to have data enabled SIM card for GPRS

Server web application

Requirements:

- Microsoft Visual Studio 2008 Professional edition
- Windows Mobile 5 or higher SDK installed including emulators
- IIS 6 configured
- SQL Server 2005 Express Edition
- Microsoft Ajax.Net installed

3. PAN mobile application

File: **Extra Utilities code.rar**

Requirements:

- Microsoft Visual Studio 2008 Professional edition
- Windows Mobile 6.1 or higher SDK installed including emulators
- PDAs (at least 4) with Bluetooth enabled and paired-up between all the devices

4. Ontologies

Includes all the ontology files used in this research implementation and analysis

File: **chapter 6 materials (ontologies, Rules).rar**

Requirements:

- Protege *or*
- Altova SemanticWorks 2008

5. XML files: Examples Rules, Plans and messages

Includes all the XML files describing rules, sample plans and messages used for Agent based interaction during the implementation and analysis of the platform

Files: **chapter 7-8 materials.rar**
chapter 5 XML files.rar

Requirements:

- Altova UModel
- Altova SemanticWorks or Protege
- Altova XMLSpy
- Notepad (or any XML editor)

Appendix K: NetLogo simulation for traffic evaluation

During the initial phase of the research, NetLogo Agent simulation tool was used in various simulation studies primarily to simulate traffic light controls. In this appendix, one of the NetLogo based simulation studies is outlined in which a traffic simulator application was implemented in NetLogo. In this application Agents were configured to perform the behaviour of traffic lights and vehicles allowing the simulation of different traffic scenarios. Each agent receives the data from the application including:

- The current the traffic lights phase. The simulator observes current active traffic lights phase that is being set to green.
- Value of the counter (for the green)
- The length of the Queue at each link waiting to be serviced. This is can be done by calculating the deference between the numbers of the vehicles entered the link and the vehicles left the link of each ln. These two numbers obtained by using sensors.
- The space available for the vehicles to file on the downstream link. (The downstream is the other side of the junction.)

By using the Traffic Simulator application the performance of the traffic was evaluated. This was done by programming a vehicle to calculate its delayed time when it is waiting in the queue, later the average delay for all vehicles can be obtained. Then the average delayed time can be compared to other simulation results and to real-time data. The simulation of simple intersection for 2 traffic lights and basic grid already exist within the tool. The enhancement of the existing simulation has been attempted in order to include some of the aspects of the traffic network model for an intersection controlled by up to 8 traffic lights.

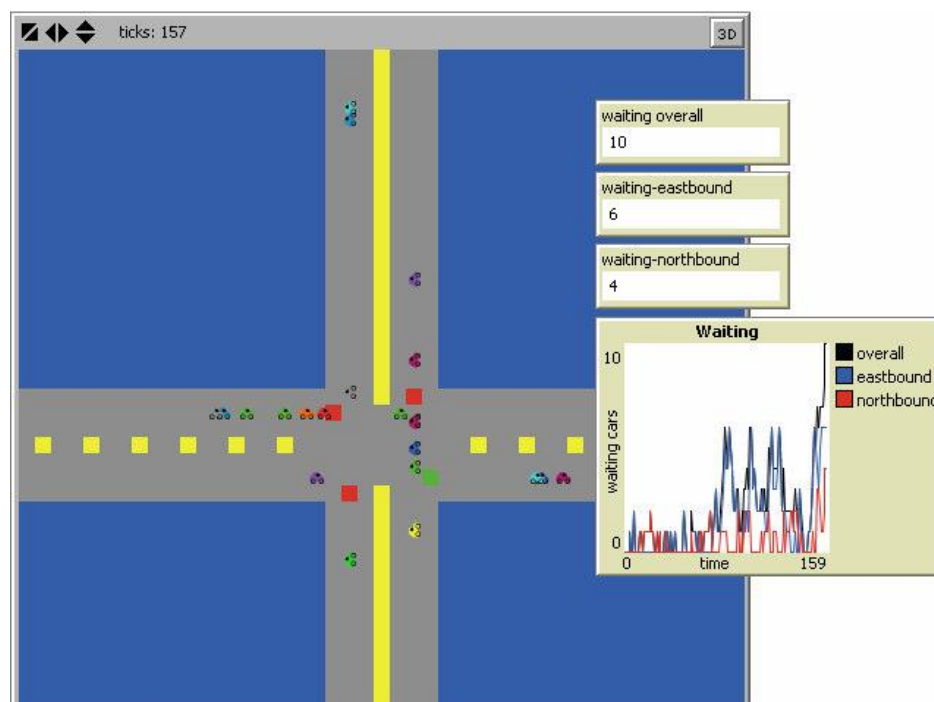


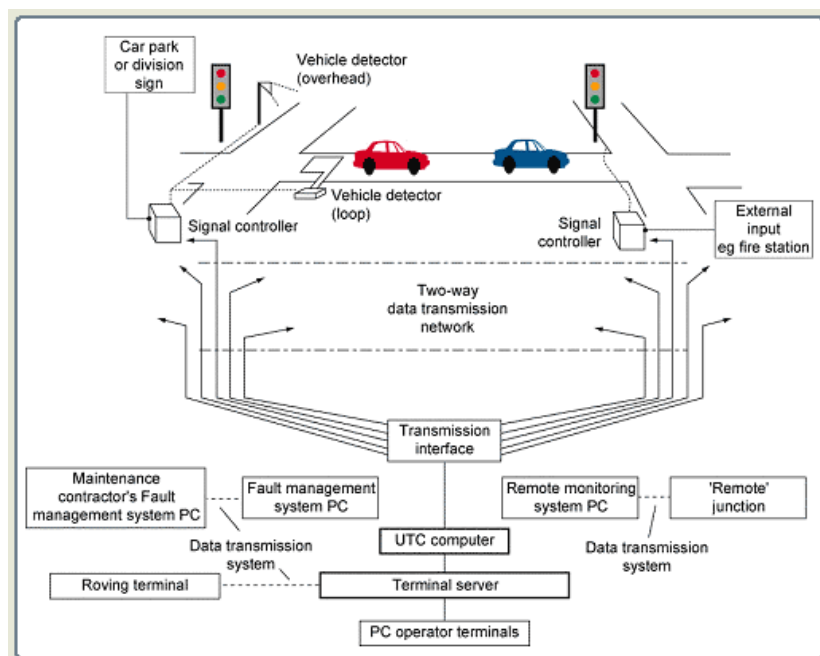
Figure: Traffic Simulator (NetLogo)

The complete code is provided in “Appendix J, CD”

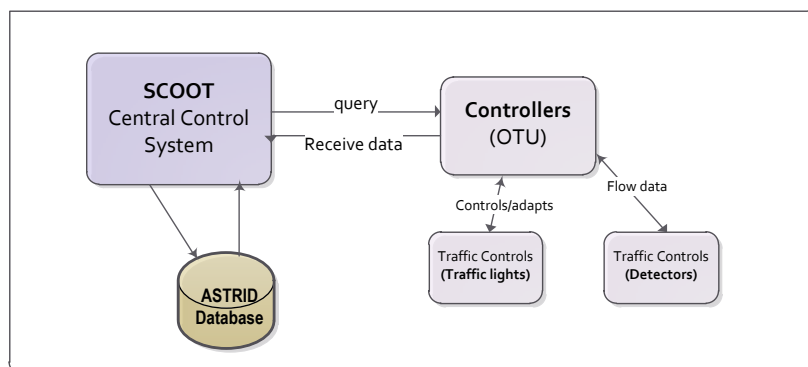
Appendix L: SCOOT data and traffic area study report

This document outlines the traffic data provided by the UTM Control Room, Coventry City Council responsible for the Transport System Management of Coventry City and surrounding areas. They use SCOOT and related add-on systems (ASTRID and INGRID) for managing traffic control systems.

SCOOT obtains the traffic flow information by calculating the data from the roadside inductive loops detectors. They come in detector packs connected to out-station transmission unit (OTU) which is the Controller with modem and connected to the SCOOT system at the central control room system. Detectors are normally required on every road link. Their location is important and they are usually positioned at the upstream end of the approach link. Inductive loops are normally used, but other methods are also available. When vehicles pass the detector, SCOOT receives the information and converts the data into its internal units and uses them to construct "Cyclic flow profiles" for each link. As an adaptive system, SCOOT depends on reliable and real-time traffic data in order to respond to changes in the flow.



SCOOT Overview (Siemens Mobility, 2009)



SCOOT System Overview (UTMC Coventry)

There are different messages to obtain data from SCOOT for example:

- M14 Link<LINK> IVL aaaa OCC bbbb LQ cccc BQ dddd EB eeee LIT <REP> f

This message is generated every four seconds for a non-faulty link.

IVL	is current interval within the profile
OCC	is the occupancy value arriving at stop-line (LPU/interval)
LQ	is the length of queue currently modelled (LPU)
BQ	is the position of back queue (LPU)
EB	exit blocked flag (0=no, 1=yes)
LIT	is four bits giving the effective state of signals in previous four seconds (1=green, 0=red)

- M37 Node <NODE> UTC a IG b GN cc Length ddd

The UTC stage lengths as measured from the stage replies.

UTC	The UTC stage
IG	The preceding inter green
GN	Green length(s)
Length	Total length = IG + green length(s)

Alternatively, data can be collected using the ASTRID, which is a software utility to access the database used to store information derived from SCOOT system such as:

- Flow: flow in vehicles per hour as modelled by SCOOT
- Flow: flow in vehicles per hour derived from detectors (best for links with one detector per lane)
- Delay: total delay in vehicles per hour
- Congestion: percentage of 4 second intervals when a detector is occupied by traffic
- Emissions estimates

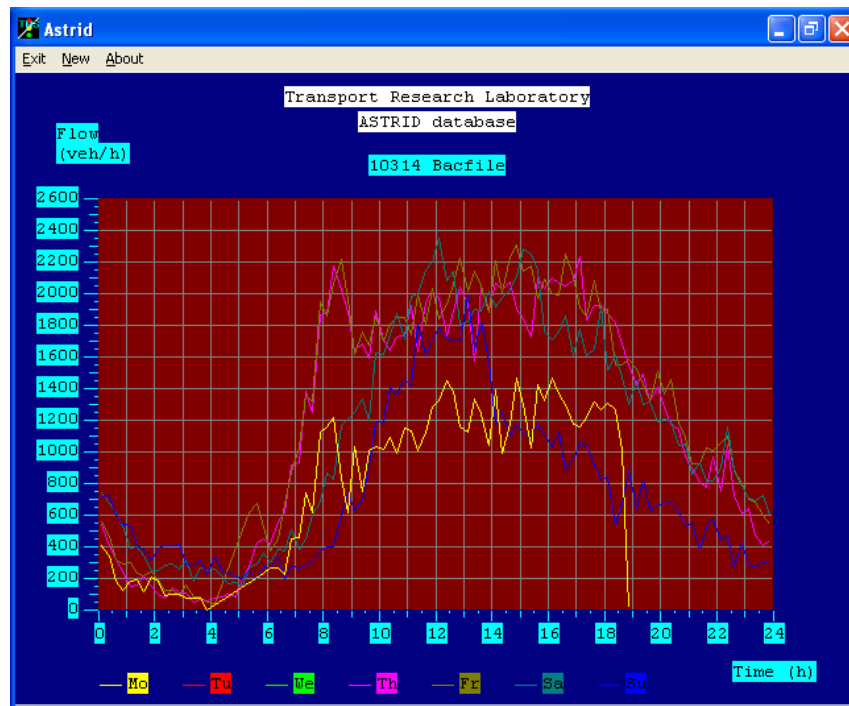
ASTRID data utility interface

The current value of any of these items is also available to the user in the form of SCOOT messages. In addition to the data directly output, ASTRID calculates and stores the following information:

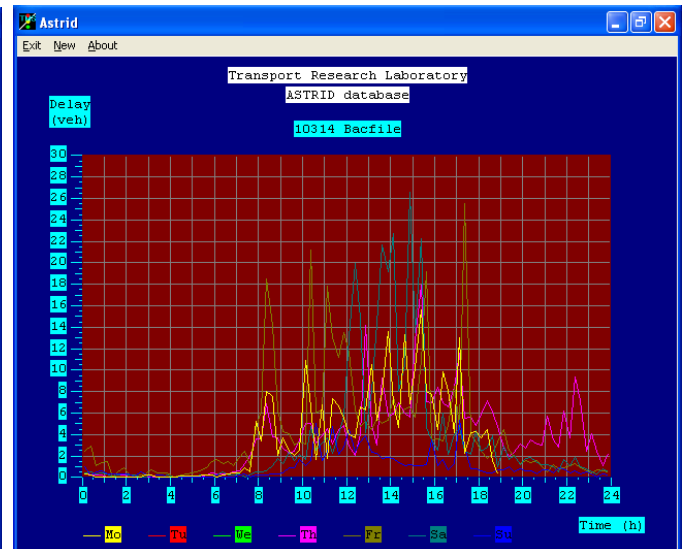
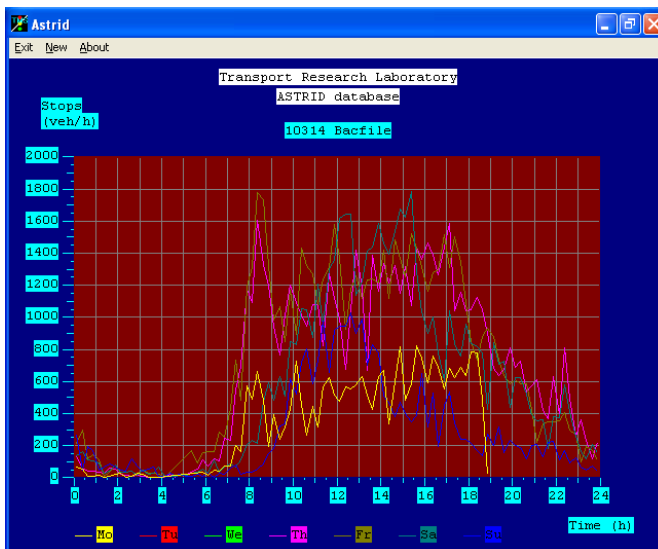
- Vehicle delay: mean delay per vehicle
- Journey time: obtained by adding vehicle delay to a pre-measured 'cruise time'
- Speed: derived from link length, cruise time and vehicle delay
- Congestion index: derived from vehicle delay and cruise time

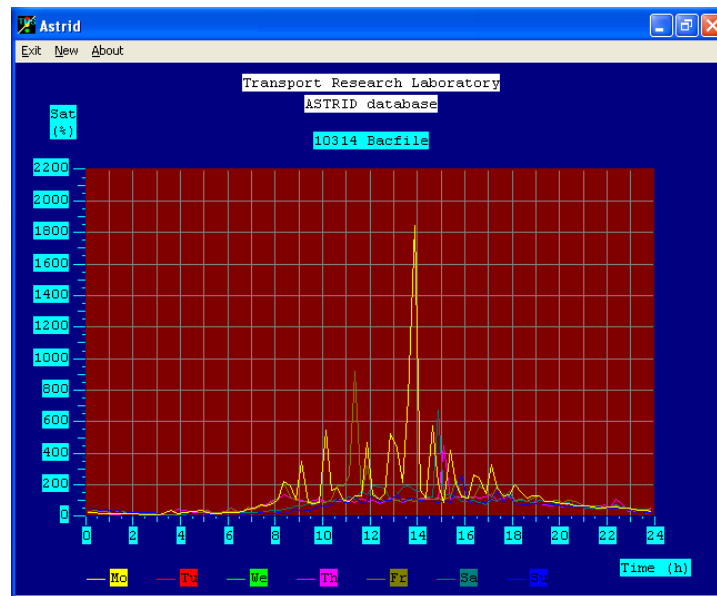
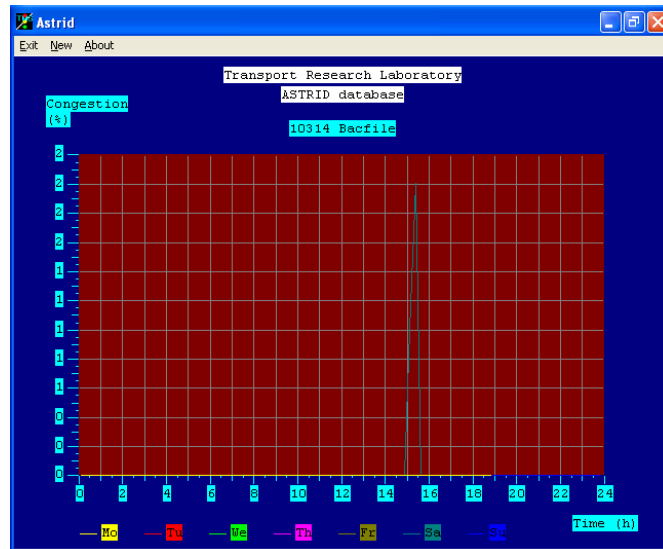
The data is available at the level of link, node, region, area or route ('route' is any pre-defined set of links). Both current and historic data is available.

Using the data obtained from ASTRID application, the following graphs were generated. For example, the following figure shows the flow for the “10314” junction over a week.



The following figures shows the Stops, Delays, Congestion and Saturation graphs for “10314” junction over a week

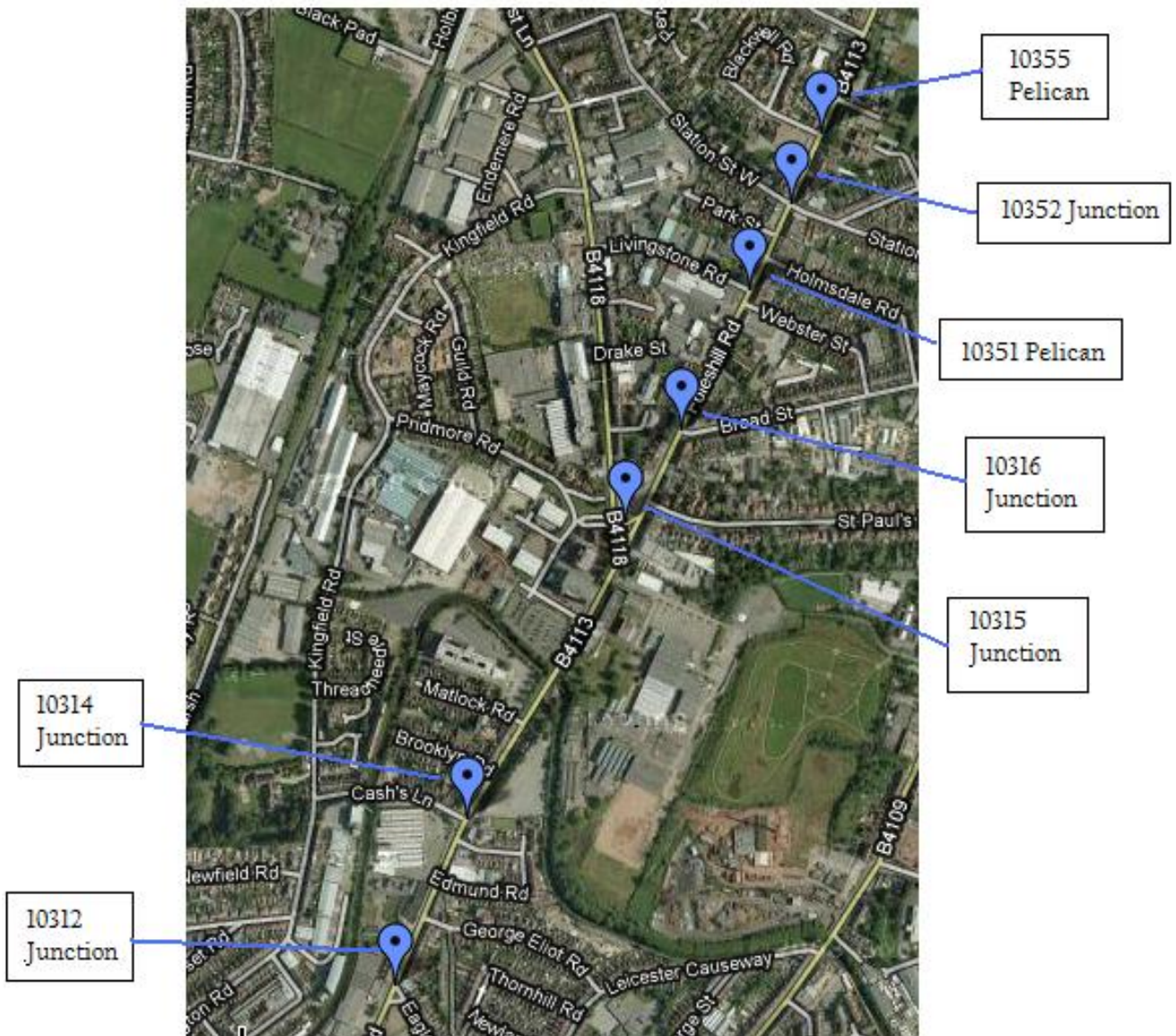




Geographical study area selection

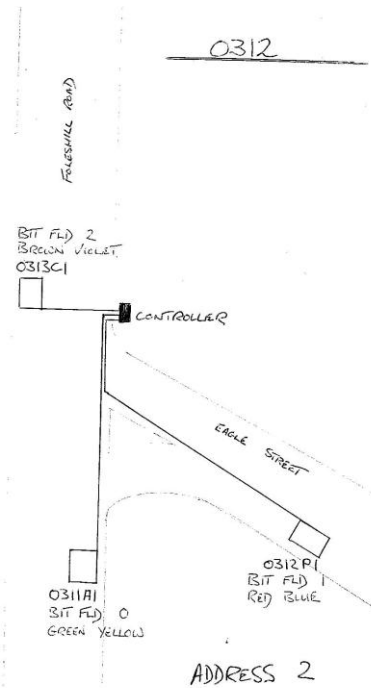
Transport Communication Centre, Coventry City Council provided extensive range of historical traffic flow data for different routes of Coventry for the purpose this research. The data obtained from SCOOT and ASTRID was assessed in order to analyse the traffic trends and select appropriate section for the simulation studies. As seen in the following figure, a section around “Foleshill Road” and associated links were focused in this research which is a major arterial connecting the centre of Coventry with the M6.

Signal Controller Number (SCN)	Foleshill Road Corridor, Coventry	Type
10311	Harnell Lane	Junction
10312	Eagle Street	Junction
10313	Edmund Road	Bus Gate
10314	Cash's Lane	Junction
10315	Lockhurst Lane	Junction
10316	Broad Street	Junction
10351	Livingstone Road	Pelican
10352	Station Street	Junction
10355	Blackwell Road	Pelican

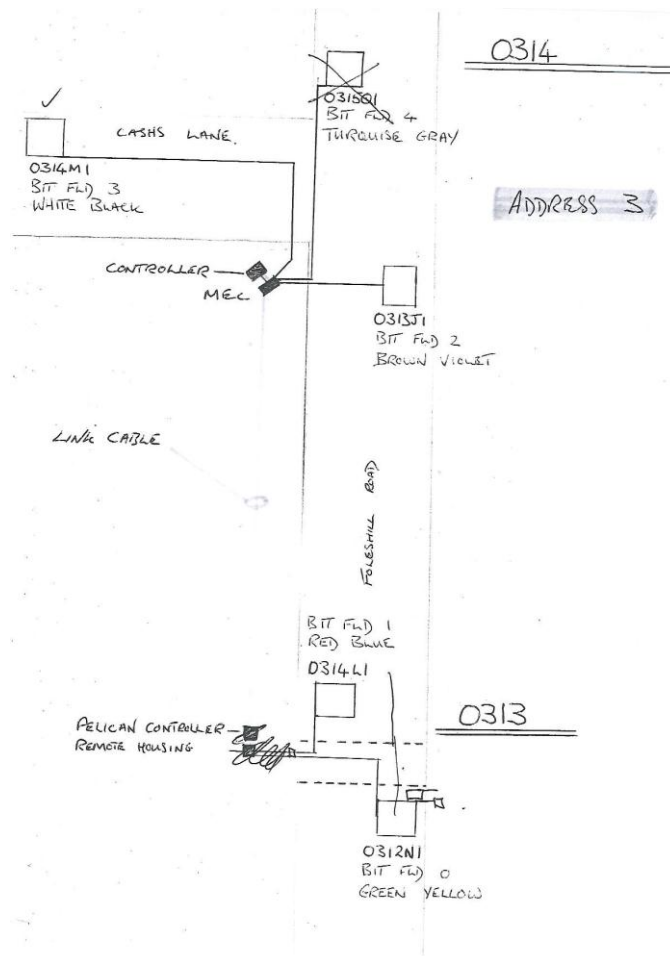


Study Area map image

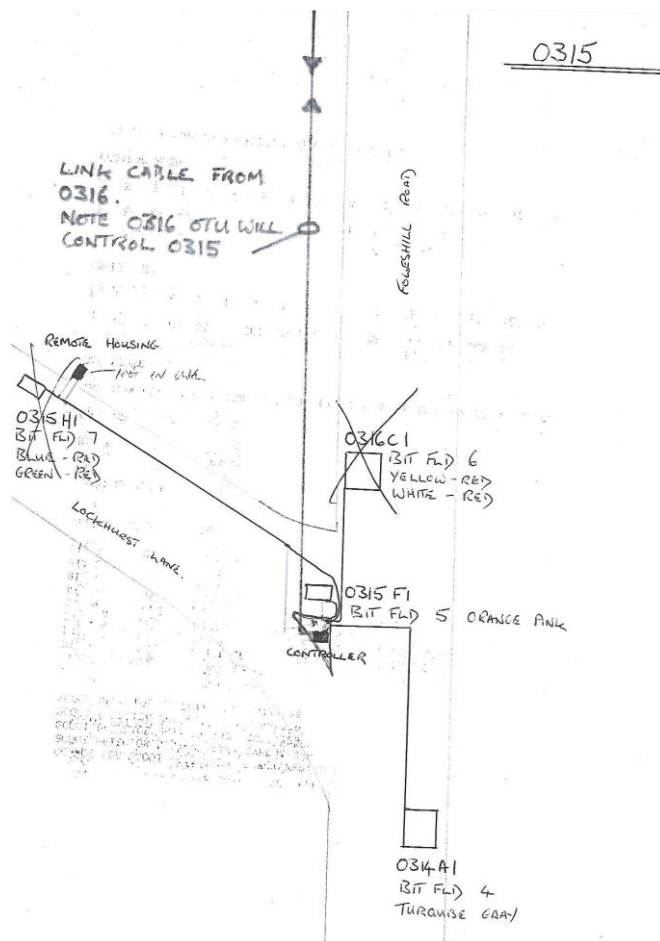
Following are the detail illustrations provided by transport Communication Centre for each SCN (in SCOOT)



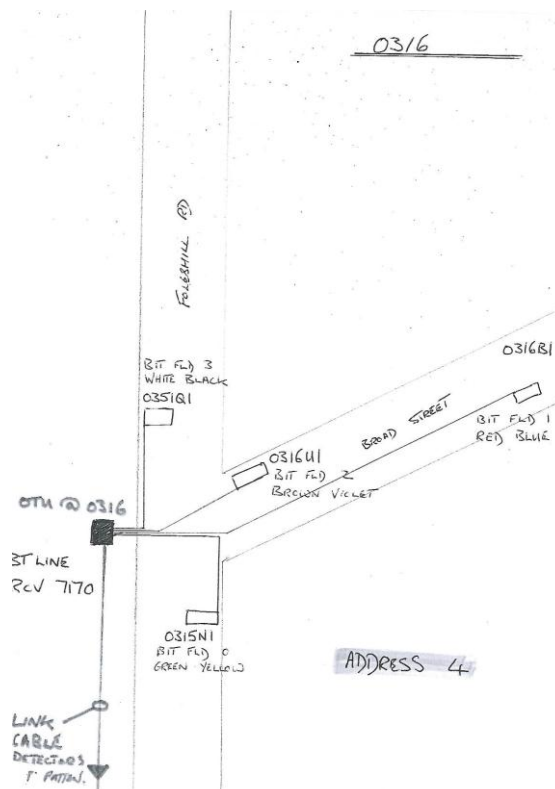
10312 Junction



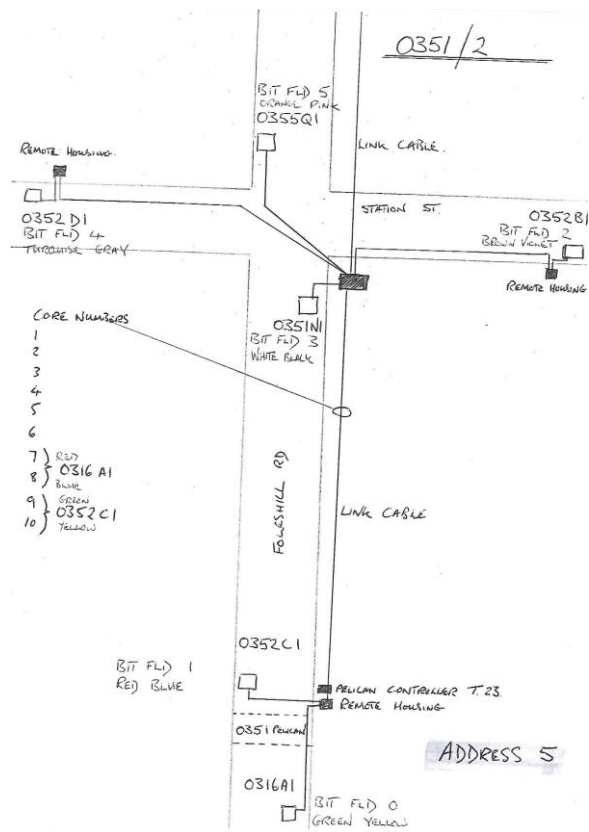
10314 Junction



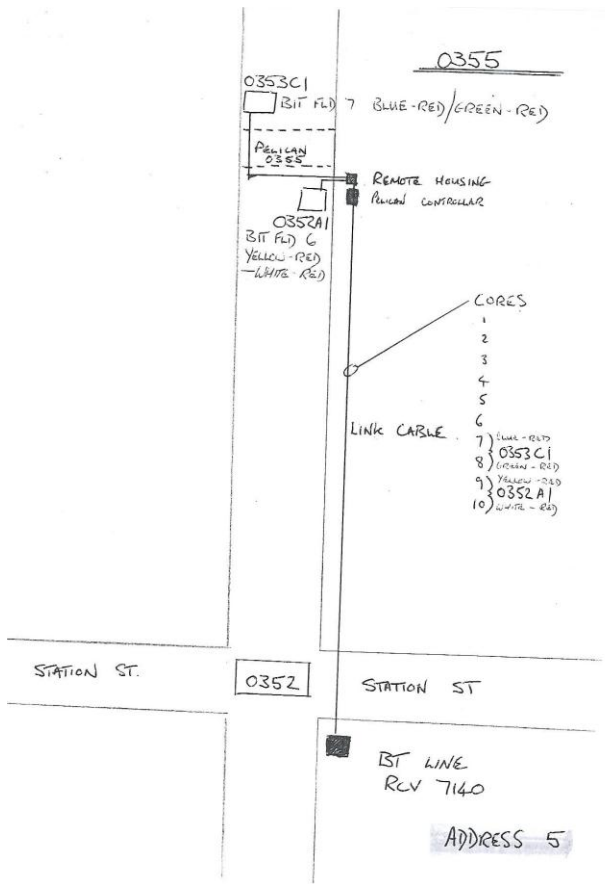
10315 Junction



10316 Junction



10351 Pelican and 10352 Junction



10355 Pelican

The data was extracted from junction/SCN 10312, 10314, 10315 and 10352, at different times, i.e. 07:00 to 10:00 and 16:00 to 19:00, and different days, i.e. Saturday to Tuesday:



Raw data (from ASTRID)

		Start		Flow Mean	Flow Count	Stops Mean	Stops Count	Delay Mean	Delay Count	Congestion Mean	Saturation Mean	Sat Count	Flow* Mean	Flow* Count	Occ* Mean	Occ* Count	Length Mean	Length Count		
Site	Day Date	Time	Time																veh/h	veh/h
10314	SA	6-Aug-2011	00:00	00:15	739	71	131	71	0.3	71	0	71	42	94	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	00:15	00:30	668	62	159	62	0.4	62	0	62	38	93	0	0	0.0	0	96	96
10314	SA	6-Aug-2011	00:30	00:45	616	71	107	71	0.2	71	0	71	35	65	0	0	0.0	0	95	101
10314	SA	6-Aug-2011	00:45	01:00	526	62	99	62	0.3	62	0	62	37	90	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	01:00	01:15	387	71	88	71	0.2	71	0	71	27	102	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	01:15	01:30	395	62	23	62	0.0	62	0	62	27	67	0	0	0.0	0	95	96
10314	SA	6-Aug-2011	01:30	01:45	337	71	37	71	0.1	71	0	71	22	54	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	01:45	02:00	244	62	71	62	0.2	62	0	62	12	97	0	0	0.0	0	95	101
10314	SA	6-Aug-2011	02:00	02:15	243	71	26	71	0.1	71	0	71	10	103	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	02:15	02:30	280	62	31	62	0.1	62	0	62	12	11	0	0	0.0	0	96	96
10314	SA	6-Aug-2011	02:30	02:45	291	71	37	71	0.0	71	0	71	14	54	0	0	0.0	0	95	101
10314	SA	6-Aug-2011	02:45	03:00	254	62	9	62	0.0	62	0	62	9	12	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	03:00	03:15	316	71	40	71	0.1	71	0	71	13	83	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	03:15	03:30	182	62	18	62	0.0	62	0	62	8	118	0	0	0.0	0	95	96
10314	SA	6-Aug-2011	03:30	03:45	259	71	23	71	0.0	71	0	71	9	79	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	03:45	04:00	248	62	60	62	0.1	62	0	62	12	90	0	0	0.0	0	95	101
10314	SA	6-Aug-2011	04:00	04:15	253	71	3	71	0.0	71	0	71	7	87	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	04:15	04:30	216	62	1	62	0.0	62	0	62	8	35	0	0	0.0	0	96	96
10314	SA	6-Aug-2011	04:30	04:45	165	71	1	71	0.0	71	0	71	0	0	0	0	0.0	0	95	101
10314	SA	6-Aug-2011	04:45	05:00	173	62	8	62	0.0	62	0	62	9	5	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	05:00	05:15	144	71	7	71	0.0	71	0	71	3	2	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	05:15	05:30	259	62	21	62	0.1	62	0	62	10	23	0	0	0.0	0	95	96
10314	SA	6-Aug-2011	05:30	05:45	285	71	30	71	0.1	71	0	71	10	56	0	0	0.0	0		

10314	SA	6-Aug-2011	12:30	12:45	2136	71	1641	71	14.8	71	0	71	170	108	0	0	0.0	0	95	101
10314	SA	6-Aug-2011	12:45	13:00	1793	62	1131	62	4.1	62	0	62	103	98	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	13:00	13:15	1822	71	1222	71	8.0	71	0	71	132	99	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	13:15	13:30	1898	62	1409	62	14.3	62	0	62	183	100	0	0	0.0	0	95	96
10314	SA	6-Aug-2011	13:30	13:45	1887	71	1435	71	21.6	71	0	71	188	100	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	13:45	14:00	2016	62	1578	62	19.1	62	0	62	156	98	0	0	0.0	0	95	101
10314	SA	6-Aug-2011	14:00	14:15	1914	71	1458	71	22.6	71	0	71	150	104	0	0	0.0	0	96	101
10314	SA	6-Aug-2011	14:15	14:30	1971	62	1388	62	7.9	62	0	62	119	97	0	0	0.0	0	97	98
10314	SA	6-Aug-2011	14:30	14:45	2022	71	1530	71	10.4	71	0	71	118	100	0	0	0.0	0	94	101
10314	SA	6-Aug-2011	14:45	15:00	2106	62	1670	62	26.5	62	0	62	664	102	0	0	0.0	0	96	96
10314	SA	6-Aug-2011	15:00	15:15	2276	71	1617	71	13.4	71	1	71	128	94	0	0	0.0	0	97	101
10314	SA	6-Aug-2011	15:15	15:30	2250	62	1781	62	22.2	62	2	62	131	105	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	15:30	15:45	2150	71	1283	71	4.6	71	0	71	113	100	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	15:45	16:00	1757	62	1030	62	3.4	62	0	62	91	95	0	0	0.0	0	95	102
10314	SA	6-Aug-2011	16:00	16:15	1703	71	894	71	2.5	71	0	71	86	104	0	0	0.0	0	96	96
10314	SA	6-Aug-2011	16:15	16:30	1758	62	999	62	5.9	62	0	62	93	101	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	16:30	16:45	1853	71	773	71	2.1	71	0	71	84	91	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	16:45	17:00	1603	62	602	62	3.9	62	0	62	70	99	0	0	0.0	0	95	96
10314	SA	6-Aug-2011	17:00	17:15	1774	71	1035	71	4.6	71	0	71	99	108	0	0	0.0	0	97	101
10314	SA	6-Aug-2011	17:15	17:30	1607	62	831	62	2.4	62	0	62	89	100	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	17:30	17:45	1650	71	754	71	2.1	71	0	71	137	97	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	17:45	18:00	1912	62	954	62	4.3	62	0	62	120	105	0	0	0.0	0	95	101
10314	SA	6-Aug-2011	18:00	18:15	1514	71	830	71	2.4	71	0	71	85	97	0	0	0.0	0	96	96
10314	SA	6-Aug-2011	18:15	18:30	1608	62	809	62	2.7	62	0	62	92	102	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	18:30	18:45	1467	71	774	71	3.8	71	0	71	90	101	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	18:45	19:00	1299	62	417	62	0.9	62	0	62	78	95	0	0	0.0	0	95	96
10314	SA	6-Aug-2011	19:00	19:15	1450	71	839	71	2.8	71	0	71	98	101	0	0	0.0	0	97	101
10314	SA	6-Aug-2011	19:15	19:30	1297	62	699	62	1.6	62	0	62	85	100	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	19:30	19:45	1312	71	719	71	2.0	71	0	71	86	103	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	19:45	20:00	1191	62	431	62	1.1	62	0	62	73	101	0	0	0.0	0	95	102
10314	SA	6-Aug-2011	20:00	20:15	1201	71	620	71	1.7	71	0	71	77	101	0	0	0.0	0	96	96
10314	SA	6-Aug-2011	20:15	20:30	1185	62	617	62	1.8	62	0	62	82	99	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	20:30	20:45	1039	71	514	71	1.3	71	0	71	65	102	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	20:45	21:00	1044	62	391	62	1.1	62	0	62	53	98	0	0	0.0	0	95	96
10314	SA	6-Aug-2011	21:00	21:15	854	71	348	71	1.1	71	0	71	47	102	0	0	0.0	0	97	101
10314	SA	6-Aug-2011	21:15	21:30	933	62	355	62	0.9	62	0	62	47	98	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	21:30	21:45	815	71	176	71	0.4	71	0	71	42	66	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	21:45	22:00	809	62	372	62	1.6	62	0	62	49	95	0	0	0.0	0	95	102
10314	SA	6-Aug-2011	22:00	22:15	894	71	369	71	1.0	71	0	71	55	104	0	0	0.0	0	96	96
10314	SA	6-Aug-2011	22:15	22:30	1155	62	575	62	1.8	62	0	62	69	100	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	22:30	22:45	882	71	409	71	0.9	71	0	71	56	102	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	22:45	23:00	776	62	255	62	0.7	62	0	62	48	99	0	0	0.0	0	95	96
10314	SA	6-Aug-2011	23:00	23:15	706	71	158	71	0.4	71	0	71	42	55	0	0	0.0	0	97	101
10314	SA	6-Aug-2011	23:15	23:30	691	62	113	62	0.2	62	0	62	41	59	0	0	0.0	0	94	102
10314	SA	6-Aug-2011	23:30	23:45	717	71	204	71	0.7	71	0	71	41	101	0	0	0.0	0	97	102
10314	SA	6-Aug-2011	23:45	24:00	599	62	148	62	0.4	62	0	62	28	99	0	0	0.0	0	95	102
10314	SU	7-Aug-2011	00:00	00:15	713	71	262	71	0.9	71	0	71	42	104	0	0	0.0	0	96	96
10314	SU	7-Aug-2011	00:15	00:30	721	62	44	62	0.1	62	0	62	39	99	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	00:30	00:45	598	71	187	71	0.4	71	0	71	31	98	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	00:45	01:00	541	62	163	62	0.5	62	0	62	37	85	0	0	0.0	0	95	96
10314	SU	7-Aug-2011	01:00	01:15	528	71	39	71	0.2	71	0	71	26	112	0	0	0.0	0	97	101
10314	SU	7-Aug-2011	01:15	01:30	407	62	59	62	0.2	62	0	62	29	101	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	01:30	01:45	382	71	82	71	0.2	71	0	71	14	105	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	01:45	02:00	313	62	58	62	0.2	62	0	62	18	15	0	0	0.0	0	95	102
10314	SU	7-Aug-2011	02:00	02:15	386	71	41	71	0.2	71	0	71	15	36	0	0	0.0	0	96	96
10314	SU	7-Aug-2011	02:15	02:30	403	62	34	62	0.0	62	0	62	16	70	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	02:30	02:45	71	110	71	0.4	71	0	71	16	108	0	0	0.0	0	94	102	
10314	SU	7-Aug-2011	02:45	03:00	415	62	65	62	0.2	62	0	62	15	105	0	0	0.0	0	95	96
10314	SU	7-Aug-2011	03:00	03:15	287	71	34	71	0.0	71	0	71	10	32	0	0	0.0	0	97	101
10314	SU	7-Aug-2011	03:15	03:30	279	62	50	62	0.1	62	0	62	12	73	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	03:30	03:45	311	71	63	71	0.1	71	0	71	13	91	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	03:45	04:00	224	62	6	62	0.0	62	0	62	0	0	0	0	0.0	0	95	102
10314	SU	7-Aug-2011	04:00	04:15	336	71	2	71	0.0	71	0	71	0	0	0	0	0.0	0	96	96
10314	SU	7-Aug-2011	04:15	04:30	244	62	22	62	0.0	62	0	62	10	62	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	04:30	04:45	231	71	8	71	0.0	71	0	71	10	13	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	04:45	05:00	205	62	15	62	0.0	62	0	62	9	24	0	0	0.0	0	95	96
10314	SU	7-Aug-2011	05:00	05:15	192	71	6	71	0.0	71	0	71	7	91	0	0	0.0	0	97	101
10314	SU	7-Aug-2011	05:15	05:30	242	62	5	62	0.0	62	0	62	3	10	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	05:30	05:45	198	71	19	71	0.0	71	0	71	8	24	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	05:45	06:00	270	62	5	62	0.0	62	0	62	13	23	0	0	0.0	0	95	102
10314	SU	7-Aug-2011	06:00	06:15	265	71	22	71	0.1	71	0	71	10	71	0	0	0.0	0	96	96
10314	SU	7-Aug-2011	06:15	06:30	329	62	0	62	0.0	62	0	62	12	28	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	06:30	06:45	197	71	23	71	0.0	71	0	71	7	9	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	06:45	07:00	275	62	14	62	0.0	62	0	62	14	24	0	0	0.0	0	95	

10314	SU	7-Aug-2011	15:45	16:00	1105	31	644	31	3.4	31	0	31	258	96	0	0	0.0	0	95	102
10314	SU	7-Aug-2011	16:00	16:15	1028	36	308	36	1.0	36	0	36	86	91	0	0	0.0	0	96	96
10314	SU	7-Aug-2011	16:15	16:30	1119	31	521	31	1.6	31	0	31	72	111	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	16:30	16:45	880	35	189	35	0.6	35	0	35	75	100	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	16:45	17:00	942	31	448	31	1.2	31	0	31	79	104	0	0	0.0	0	95	96
10314	SU	7-Aug-2011	17:00	17:15	1053	36	527	36	5.2	36	0	36	112	96	0	0	0.0	0	97	101
10314	SU	7-Aug-2011	17:15	17:30	1041	31	338	31	2.7	31	0	31	151	99	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	17:30	17:45	916	35	236	35	0.7	35	0	35	84	101	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	17:45	18:00	817	31	236	31	0.7	31	0	31	142	100	0	0	0.0	0	95	102
10314	SU	7-Aug-2011	18:00	18:15	834	36	210	36	0.5	36	0	36	76	101	0	0	0.0	0	96	96
10314	SU	7-Aug-2011	18:15	18:30	529	31	164	31	0.3	31	0	31	65	102	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	18:30	18:45	672	35	132	35	0.3	35	0	35	66	101	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	18:45	19:00	882	31	265	31	0.6	31	0	31	73	99	0	0	0.0	0	95	96
10314	SU	7-Aug-2011	19:00	19:15	636	36	195	36	0.6	36	0	36	74	100	0	0	0.0	0	97	101
10314	SU	7-Aug-2011	19:15	19:30	823	31	313	31	0.9	31	0	31	73	97	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	19:30	19:45	633	35	151	35	0.4	35	0	35	66	103	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	19:45	20:00	658	31	226	31	0.7	31	0	31	65	101	0	0	0.0	0	95	102
10314	SU	7-Aug-2011	20:00	20:15	680	36	197	36	0.5	36	0	36	59	95	0	0	0.0	0	96	96
10314	SU	7-Aug-2011	20:15	20:30	671	31	181	31	0.5	31	0	31	61	101	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	20:30	20:45	619	35	115	35	0.2	35	0	35	55	88	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	20:45	21:00	531	31	194	31	0.6	31	0	31	52	113	0	0	0.0	0	95	96
10314	SU	7-Aug-2011	21:00	21:15	545	36	204	36	0.6	36	0	36	54	101	0	0	0.0	0	97	101
10314	SU	7-Aug-2011	21:15	21:30	381	31	127	31	0.3	31	0	31	54	100	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	21:30	21:45	523	35	216	35	0.8	35	0	35	55	101	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	21:45	22:00	571	31	219	31	0.7	31	0	31	46	98	0	0	0.0	0	95	102
10314	SU	7-Aug-2011	22:00	22:15	437	36	108	36	0.3	36	0	36	46	102	0	0	0.0	0	96	96
10314	SU	7-Aug-2011	22:15	22:30	468	31	165	31	0.5	31	0	31	49	101	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	22:30	22:45	271	35	87	35	0.2	35	0	35	45	97	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	22:45	23:00	419	31	120	31	0.4	31	0	31	38	96	0	0	0.0	0	95	96
10314	SU	7-Aug-2011	23:00	23:15	278	36	63	36	0.2	36	0	36	19	96	0	0	0.0	0	97	101
10314	SU	7-Aug-2011	23:15	23:30	268	31	40	31	0.0	31	0	31	24	68	0	0	0.0	0	94	102
10314	SU	7-Aug-2011	23:30	23:45	304	35	66	35	0.2	35	0	35	18	105	0	0	0.0	0	97	102
10314	SU	7-Aug-2011	23:45	24:00	290	31	38	31	0.1	31	0	31	17	101	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	00:00	00:15	406	36	59	36	0.2	36	0	36	17	97	0	0	0.0	0	96	96
10314	MO	8-Aug-2011	00:15	00:30	341	31	51	31	0.1	31	0	31	16	102	0	0	0.0	0	97	102
10314	MO	8-Aug-2011	00:30	00:45	177	35	4	35	0.0	35	0	35	10	51	0	0	0.0	0	94	102
10314	MO	8-Aug-2011	00:45	01:00	121	31	5	31	0.0	31	0	31	9	69	0	0	0.0	0	95	96
10314	MO	8-Aug-2011	01:00	01:15	182	36	10	36	0.0	36	0	36	11	23	0	0	0.0	0	97	101
10314	MO	8-Aug-2011	01:15	01:30	189	31	0	31	0.0	31	0	31	0	0	0	0	0.0	0	94	102
10314	MO	8-Aug-2011	01:30	01:45	115	35	4	35	0.0	35	0	35	0	0	0	0	0.0	0	97	102
10314	MO	8-Aug-2011	01:45	02:00	202	31	18	31	0.0	31	0	31	11	87	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	02:00	02:15	176	36	23	36	0.0	36	0	36	9	52	0	0	0.0	0	96	96
10314	MO	8-Aug-2011	02:15	02:30	89	31	0	31	0.0	31	0	31	5	49	0	0	0.0	0	97	102
10314	MO	8-Aug-2011	02:30	02:45	98	35	4	35	0.0	35	0	35	0	0	0	0	0.0	0	94	102
10314	MO	8-Aug-2011	02:45	03:00	96	31	25	31	0.1	31	0	31	5	35	0	0	0.0	0	95	96
10314	MO	8-Aug-2011	03:00	03:15	75	36	18	36	0.1	36	0	36	6	29	0	0	0.0	0	97	101
10314	MO	8-Aug-2011	03:15	03:30	70	31	0	31	0.0	31	0	31	0	0	0	0	0.0	0	94	102
10314	MO	8-Aug-2011	03:30	03:45	69	35	0	35	0.0	35	0	35	32	24	0	0	0.0	0	97	102
10314	MO	8-Aug-2011	03:45	04:00	0	4	0	4	0.0	4	0	4	10	33	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	04:00	04:15	0	0	0	0	0.0	0	0	0	0	0	0	0	0.0	0	96	96
10314	MO	8-Aug-2011	04:15	04:30	0	0	0	0	0.0	0	0	0	0	0	0	0	0.0	0	97	102
10314	MO	8-Aug-2011	04:30	04:45	0	0	0	0	0.0	0	0	0	0	0	0	0	0.0	0	94	102
10314	MO	8-Aug-2011	04:45	05:00	0	0	0	0	0.0	0	0	33	29	0	0	0.0	0	95	96	
10314	MO	8-Aug-2011	05:00	05:15	0	0	0	0	0.0	0	0	17	50	0	0	0.0	0	97	101	
10314	MO	8-Aug-2011	05:15	05:30	0	0	0	0	0.0	0	0	11	113	0	0	0.0	0	94	102	
10314	MO	8-Aug-2011	05:30	05:45	0	0	0	0	0.0	0	0	14	23	0	0	0.0	0	97	102	
10314	MO	8-Aug-2011	05:45	06:00	228	27	31	27	0.1	27	0	27	19	94	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	06:00	06:15	261	36	11	36	0.0	36	0	36	20	35	0	0	0.0	0	96	96
10314	MO	8-Aug-2011	06:15	06:30	263	31	43	31	0.1	31	0	31	19	109	0	0	0.0	0	97	102
10314	MO	8-Aug-2011	06:30	06:45	224	35	38	35	0.2	35	0	35	27	99	0	0	0.0	0	94	102
10314	MO	8-Aug-2011	06:45	07:00	443	31	67	31	0.3	31	0	31	37	99	0	0	0.0	0	95	96
10314	MO	8-Aug-2011	07:00	07:15	451	36	65	36	0.3	36	0	36	43	109	0	0	0.0	0	95	106
10314	MO	8-Aug-2011	07:15	07:30	735	31	194	31	0.8	31	0	31	67	99	0	0	0.0	0	98	99
10314	MO	8-Aug-2011	07:30	07:45	611	44	158	44	0.4	44	0	44	58	97	0	0	0.0	0	97	98
10314	MO	8-Aug-2011	07:45	08:00	1118	31	568	31	5.1	31	0	31	74	97	0	0	0.0	0	99	103
10314	MO	8-Aug-2011	08:00	08:15	1157	36	487	36	3.3	36	0	36	105	106	0	0	0.0	0	98	99
10314	MO	8-Aug-2011	08:15	08:30	1209	31	658	31	7.9	31	0	31	213	101	0	0	0.0	0	93	101
10314	MO	8-Aug-2011	08:30	08:45	826	54	476	54	7.5	54	0	54	184	94	0	0	0.0	0	97	102
10314	MO	8-Aug-2011	08:45	09:00	611	44	188	44	2.0	44	0	44	101	104	0	0	0.0	0	98	98
10314	MO	8-Aug-2011	09:00	09:15	1028	36	390	36	3.6	36	0	36	344	99	0	0	0.0	0	93	101
10314	MO	8-Aug-2011	09:15	09:30	747	45	236	45	2.4	45	0	45	83	96	0	0	0.0	0	98	101
10314	MO	8-Aug-2011	09:30	09:45	1007	35	312	35	2.0	35	0	35	74	104	0	0	0.0	0	97	97
10314	MO	8-Aug-2011	09:45	10:00	1031	31	420	31	2.5	31	0	31	82	96	0	0	0.0	0	92	99
10314	MO	8-Aug-2011	10:00	10:15	1012	36	720	36	10.8	36	0	36	536	105	0	0	0.0	0	98	102
10314	MO	8-Aug-2011	10:15	10:30	1085	31	448	31	6.5	31	0	31	162	90	0	0	0.0	0		

10314	MO	8-Aug-2011	19:00	19:15	0	0	0	0	0.0	0	0	0	121	95	0	0	0.0	0	97	101
10314	MO	8-Aug-2011	19:15	19:30	0	0	0	0	0.0	0	0	0	89	101	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	19:30	19:45	0	0	0	0	0.0	0	0	0	88	103	0	0	0.0	0	96	102
10314	MO	8-Aug-2011	19:45	20:00	0	0	0	0	0.0	0	0	0	82	100	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	20:00	20:15	0	0	0	0	0.0	0	0	0	71	96	0	0	0.0	0	96	96
10314	MO	8-Aug-2011	20:15	20:30	0	0	0	0	0.0	0	0	0	77	103	0	0	0.0	0	96	102
10314	MO	8-Aug-2011	20:30	20:45	0	0	0	0	0.0	0	0	0	63	100	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	20:45	21:00	0	0	0	0	0.0	0	0	0	60	96	0	0	0.0	0	96	96
10314	MO	8-Aug-2011	21:00	21:15	0	0	0	0	0.0	0	0	0	58	103	0	0	0.0	0	97	101
10314	MO	8-Aug-2011	21:15	21:30	0	0	0	0	0.0	0	0	0	53	100	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	21:30	21:45	0	0	0	0	0.0	0	0	0	48	103	0	0	0.0	0	96	102
10314	MO	8-Aug-2011	21:45	22:00	0	0	0	0	0.0	0	0	0	43	100	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	22:00	22:15	0	0	0	0	0.0	0	0	0	50	96	0	0	0.0	0	96	96
10314	MO	8-Aug-2011	22:15	22:30	0	0	0	0	0.0	0	0	0	53	103	0	0	0.0	0	96	102
10314	MO	8-Aug-2011	22:30	22:45	0	0	0	0	0.0	0	0	0	43	100	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	22:45	23:00	0	0	0	0	0.0	0	0	0	45	96	0	0	0.0	0	96	96
10314	MO	8-Aug-2011	23:00	23:15	0	0	0	0	0.0	0	0	0	37	103	0	0	0.0	0	97	101
10314	MO	8-Aug-2011	23:15	23:30	0	0	0	0	0.0	0	0	0	29	100	0	0	0.0	0	95	102
10314	MO	8-Aug-2011	23:30	23:45	0	0	0	0	0.0	0	0	0	31	103	0	0	0.0	0	96	102
10314	MO	8-Aug-2011	23:45	24:00	0	0	0	0	0.0	0	0	0	24	100	0	0	0.0	0	95	102

Further data files containing historical traffic data are provided in “Appendix J, CD”

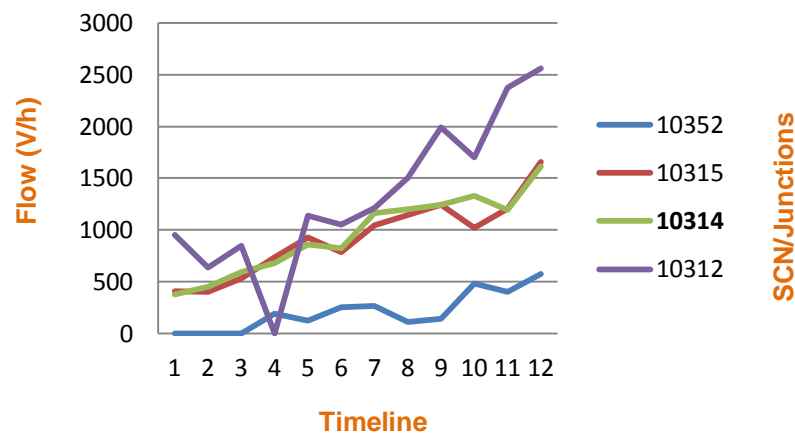
Data analysis

Legend to read charts:

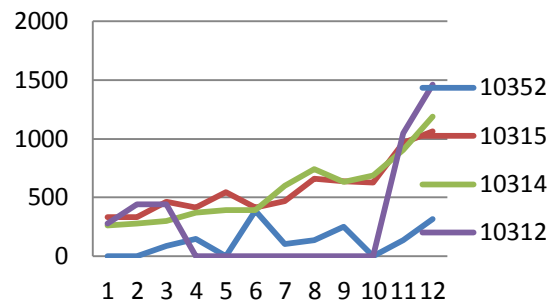
GREEN box means an increase of the flow in relation to the previous box.
RED box means a decrease of the flow in relation to the previous box.
EMPTY box indicates that the system does not collect data at this time.

The first set of times is from 07:00 to 10:00

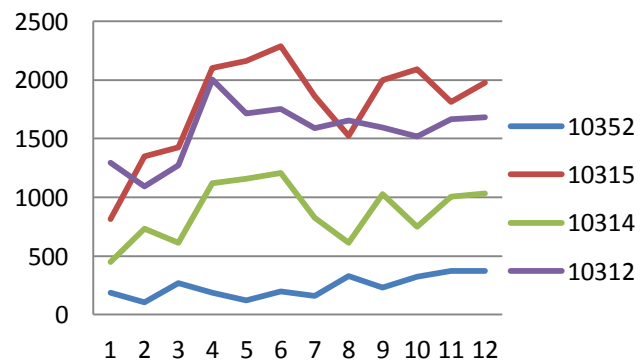
		10352	10315	10314	10312
SATURDAY	07:00 07:15		406	377	954
	07:15 07:30		405	450	636
	07:30 07:45		530	595	848
	07:45 08:00	192	735	679	
	08:00 08:15	126	926	862	1136
	08:15 08:30	252	783	822	1049
	08:30 08:45	266	1048	1166	1212
	08:45 09:00	115	1144	1203	1504
	09:00 09:15	141	1241	1243	1990
	09:15 09:30	484	1018	1330	1700
	09:30 09:45	405	1208	1195	2373
	09:45 10:00	577	1659	1617	2557



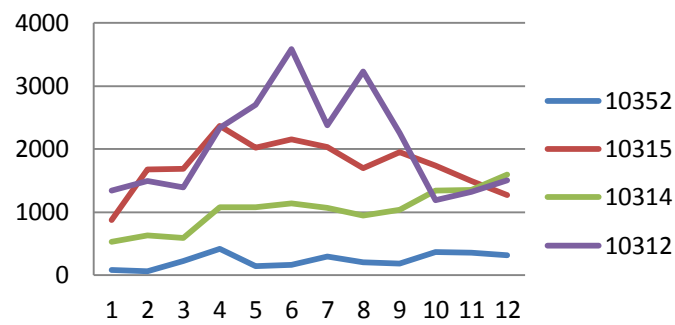
SUNDAY	07:00 07:15		332	261	282
	07:15 07:30		333	277	442
	07:30 07:45	87	467	303	443
	07:45 08:00	150	417	371	
	08:00 08:15		548	396	
	08:15 08:30	389	418	393	
	08:30 08:45	105	471	599	
	08:45 09:00	139	659	741	
	09:00 09:15	251	641	631	
	09:15 09:30		626	689	
	09:30 09:45	136	965	904	1046
	09:45 10:00	320	1062	1188	1460



MONDAY	07:00 07:15	186	815	451	1292
	07:15 07:30	104	1351	735	1093
	07:30 07:45	268	1427	611	1270
	07:45 08:00	185	2099	1118	2005
	08:00 08:15	121	2162	1157	1714
	08:15 08:30	197	2289	1209	1752
	08:30 08:45	159	1864	826	1590
	08:45 09:00	331	1523	611	1654
	09:00 09:15	230	2000	1028	1595
	09:15 09:30	323	2088	747	1518
	09:30 09:45	373	1814	1007	1666
	09:45 10:00	373	1975	1031	1679

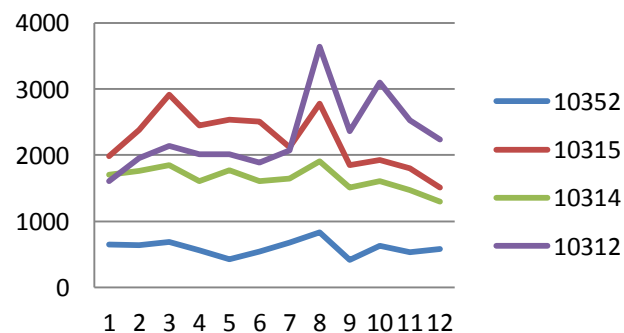


TUESDAY	07:00 07:15	83	873	535	1348
	07:15 07:30	70	1673	633	1499
	07:30 07:45	226	1686	596	1397
	07:45 08:00	421	2367	1081	2332
	08:00 08:15	147	2026	1084	2700
	08:15 08:30	164	2158	1138	3582
	08:30 08:45	296	2031	1071	2374
	08:45 09:00	208	1702	951	3231
	09:00 09:15	187	1955	1039	2254
	09:15 09:30	370	1741	1339	1194
	09:30 09:45	364	1500	1354	1322
	09:45 10:00	322	1268	1597	1509

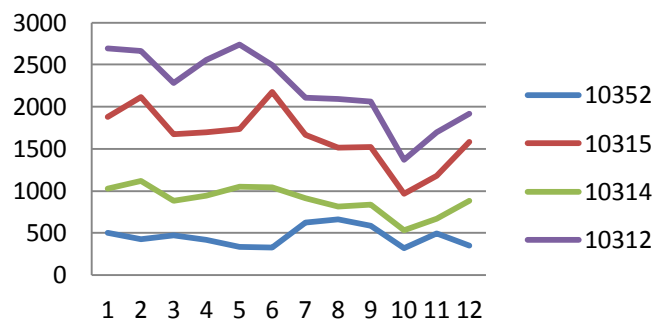


The first set of times is from 16:00 to 19:00

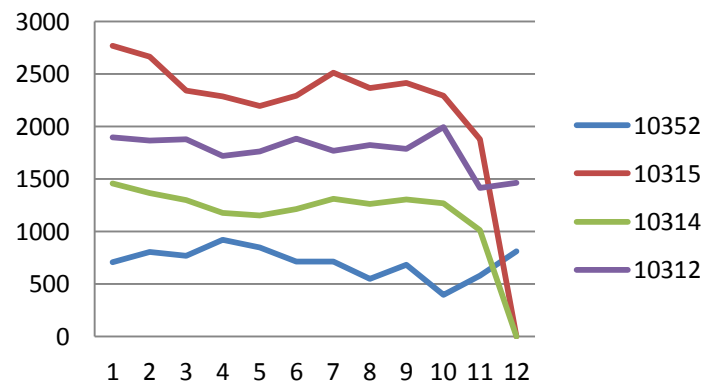
SATURDAY	16:00 16:15	650	1982	1703	1605
	16:15 16:30	635	2380	1758	1955
	16:30 16:45	688	2912	1853	2140
	16:45 17:00	561	2451	1603	2012
	17:00 17:15	425	2541	1774	2016
	17:15 17:30	540	2509	1607	1890
	17:30 17:45	682	2116	1650	2070
	17:45 18:00	835	2775	1912	3637
	18:00 18:15	417	1848	1514	2361
	18:15 18:30	625	1929	1608	3096
	18:30 18:45	533	1796	1467	2527
	18:45 19:00	583	1511	1299	2233



SUNDAY	16:00 16:15	499	1882	1028	2695
	16:15 16:30	427	2114	1119	2667
	16:30 16:45	470	1675	880	2282
	16:45 17:00	419	1694	942	2557
	17:00 17:15	336	1733	1053	2742
	17:15 17:30	325	2175	1041	2495
	17:30 17:45	626	1668	916	2108
	17:45 18:00	659	1512	817	2094
	18:00 18:15	586	1525	834	2064
	18:15 18:30	318	963	529	1366
	18:30 18:45	495	1177	672	1698
	18:45 19:00	347	1581	882	1916

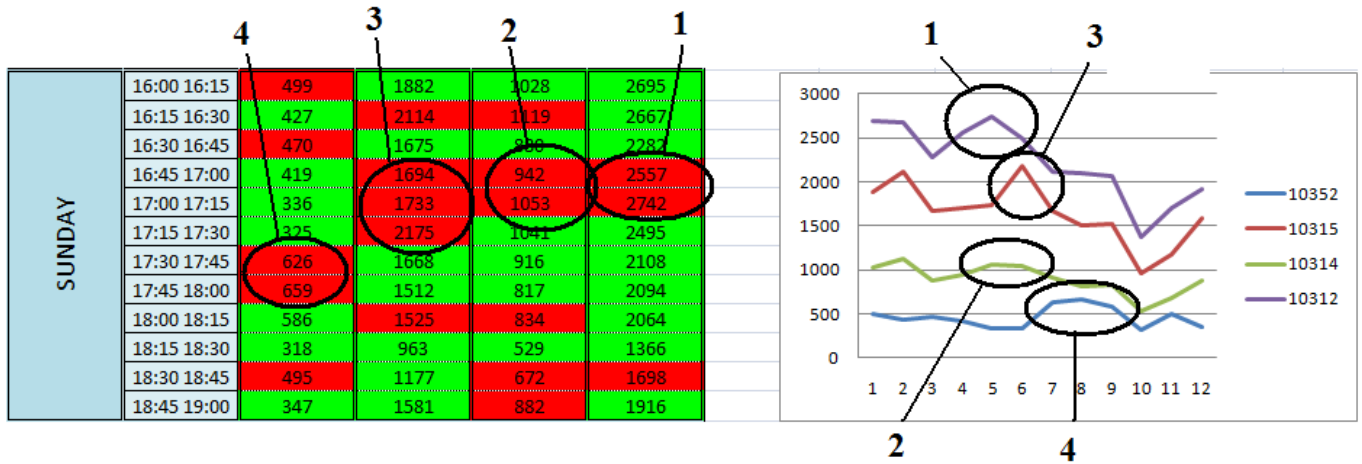


MONDAY	16:00 16:15	707	2766	1460	1893
	16:15 16:30	806	2664	1366	1863
	16:30 16:45	772	2340	1299	1875
	16:45 17:00	919	2283	1175	1722
	17:00 17:15	850	2195	1152	1759
	17:15 17:30	714	2290	1212	1881
	17:30 17:45	716	2513	1309	1767
	17:45 18:00	551	2364	1260	1824
	18:00 18:15	685	2416	1306	1785
	18:15 18:30	395	2292	1266	1992
	18:30 18:45	579	1879	1015	1415
	18:45 19:00	814			1465



The first set of times is from 07:00 to 10:00

The curves are similar for the junctions 10314 & 10315 on Saturday and Sunday. However, on Monday and Tuesday some peaks were usually observed between 07:30 and 08:00 representing the morning rush hour traffic to the city centre.

The first set of times is from 16:00 to 19:00

The data from Saturday shows three phases of increase, with a high peak at around 17:45/18:00 for all junctions. The delay can be observed as the traffic that represents the peak was moving from the junction 10312 (1) to the junction 10352 (4). The red boxes indicate that the increase is shifting. Based on the observations of these curves, there are some peaks that can be detected because there are delays between the junctions.

Appendix M: Schema XSD – Agent Message Structure

element Agent-Message

diagram						
properties	content complex					
children	Sender-List Recipient-List Semantic-Content Conditions					
attributes	Name	Type	Use	Default	Fixed	annotation
	Message-ID	derived by: xs:ID	required			
	Message-Type	derived by: xs:string	required			
	Instruction-Type	derived by: xs:string	required			
	Conversation-ID	xs:string	required			
	TimeStamp	xs:dateTime	required			
	Priority	xs:string	optional	Normal		

	Expires derived by: optional xs:dateTime Acknowledgement xs:boolean
source	<pre> <xs:element name="Agent-Message"> <xs:complexType> <xs:all> <xs:element name="Sender-List"> <xs:complexType> <xs:sequence> <xs:element name="Agent" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="ID" type="xs:ID" use="required"/> <xs:attribute name="IsCurrent-Owner" type="xs:boolean" use="required"/> <xs:attribute name="IsOrignator" type="xs:boolean"/> <xs:attribute name="Agent-Type" use="optional"/> <xs:attribute name="Agency-Number"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> <xs:element name="Recipient-List"> <xs:complexType> <xs:sequence> <xs:element name="Agent" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="ID" type="xs:ID" use="required"/> <xs:attribute name="IsMandatory" type="xs:string" use="optional"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> <xs:element name="Semantic-Content" minOccurs="0"> <xs:complexType mixed="false"> <xs:sequence> <xs:element name="Ontologies-List" minOccurs="0"> <xs:complexType> <xs:sequence> <xs:element name="Ontology" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </xs:all> </xs:complexType> </xs:element> </pre>

```

        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Context-Expression-Value" minOccurs="0">
  <xs:complexType mixed="true">
    <xs:all minOccurs="0"/>
  </xs:complexType>
</xs:element>
<xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="Use-Ontology" type="xs:string" use="optional"/>
<xs:anyAttribute namespace="##any"/>
</xs:complexType>
</xs:element>
<xs:element name="Conditions" minOccurs="0">
  <xs:complexType>
    <xs:choice>
      <xs:element name="When" type="xs:string" minOccurs="0"/>
      <xs:element name="If" type="xs:string" minOccurs="0"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
</xs:all>
<xs:attribute name="Message-ID" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:ID">
      <xs:whiteSpace value="collapse"/>
      <xs:maxLength value="30"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="Message-Type" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Request"/>
      <xs:enumeration value="Reply"/>
      <xs:enumeration value="Cancel"/>
      <xs:enumeration value="Forward-Request"/>
      <xs:enumeration value="Inform"/>
      <xs:enumeration value="Forward-Inform"/>
      <xs:enumeration value="Subscribe"/>
      <xs:enumeration value="Register"/>
      <xs:enumeration value="Propose"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="Instruction-Type" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Action"/>
      <xs:enumeration value="Query"/>
      <xs:enumeration value="Refuse"/>
      <xs:enumeration value="Failure"/>
      <xs:enumeration value="Response-Result"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

```

	<pre> <xs:enumeration value="Agree"/> <xs:enumeration value="Acknowledgement"/> <xs:enumeration value="UnSubscribe"/> <xs:enumeration value="UnRegister"/> </xs:restriction> </xs:simpleType> </xs:attribute> <xs:attribute name="Conversation-ID" type="xs:string" use="required"/> <xs:attribute name="TimeStamp" type="xs:dateTime" use="required"/> <xs:attribute name="Priority" type="xs:string" use="optional" default="Normal"/> <xs:attribute name="Expires" use="optional"> <xs:simpleType> <xs:restriction base="xs:dateTime"> <xs:pattern value=""/> </xs:restriction> </xs:simpleType> </xs:attribute> <xs:attribute name="Acknowledgement" type="xs:boolean"/> </xs:complexType> </xs:element> </pre>
--	---

attribute **Agent-Message/@Message-ID**

type	restriction of xs:ID
properties	isRef 0 use required
facets	maxLength 30 whiteSpace collapse
source	<pre> <xs:attribute name="Message-ID" use="required"> <xs:simpleType> <xs:restriction base="xs:ID"> <xs:whiteSpace value="collapse"/> <xs:maxLength value="30"/> </xs:restriction> </xs:simpleType> </xs:attribute> </pre>

attribute **Agent-Message/@Message-Type**

type	restriction of xs:string
properties	isRef 0 use required
facets	<pre> enumeration Request enumeration Reply enumeration Cancel enumeration Forward-Request enumeration Inform enumeration Forward-Inform enumeration Subscribe enumeration Register enumeration Propose </pre>
source	<pre> <xs:attribute name="Message-Type" use="required"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="Request"/> </pre>

	<pre> <xs:enumeration value="Reply"/> <xs:enumeration value="Cancel"/> <xs:enumeration value="Forward-Request"/> <xs:enumeration value="Inform"/> <xs:enumeration value="Forward-Inform"/> <xs:enumeration value="Subscribe"/> <xs:enumeration value="Register"/> <xs:enumeration value="Propose"/> </xs:restriction> </xs:simpleType> </xs:attribute> </pre>
--	---

attribute **Agent-Message/@Instruction-Type**

type	restriction of xs:string
properties	isRef 0 use required
facets	enumeration Action enumeration Query enumeration Refuse enumeration Failure enumeration Response-Result enumeration Agree enumeration Acknowledgment enumeration UnSubscribe enumeration UnRegister
source	<pre> <xs:attribute name="Instruction-Type" use="required"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="Action"/> <xs:enumeration value="Query"/> <xs:enumeration value="Refuse"/> <xs:enumeration value="Failure"/> <xs:enumeration value="Response-Result"/> <xs:enumeration value="Agree"/> <xs:enumeration value="Acknowledgment"/> <xs:enumeration value="UnSubscribe"/> <xs:enumeration value="UnRegister"/> </xs:restriction> </xs:simpleType> </xs:attribute> </pre>

attribute **Agent-Message/@Conversation-ID**

type	xs:string
properties	isRef 0 use required
source	<pre> <xs:attribute name="Conversation-ID" type="xs:string" use="required"/> </pre>

attribute **Agent-Message/@TimeStamp**

type	xs:dateTime
properties	isRef 0 use required

source	<code><xs:attribute name="TimeStamp" type="xs:dateTime" use="required"/></code>
--------	---

attribute **Agent-Message/@Priority**

type	xs:string
properties	isRef 0 default Normal use optional
source	<code><xs:attribute name="Priority" type="xs:string" use="optional" default="Normal"/></code>

attribute **Agent-Message/@Expires**

type	restriction of xs:dateTime
properties	isRef 0 use optional
facets	pattern
source	<pre> <xs:attribute name="Expires" use="optional"> <xs:simpleType> <xs:restriction base="xs:dateTime"> <xs:pattern value=""/> </xs:restriction> </xs:simpleType> </xs:attribute> </pre>

attribute **Agent-Message/@Acknowledgement**

type	xs:boolean
properties	isRef 0
source	<code><xs:attribute name="Acknowledgement" type="xs:boolean"/></code>

element **Agent-Message/Sender-List**

diagram	
properties	isRef 0 content complex
children	Agent
source	<pre> <xs:element name="Sender-List"> <xs:complexType> <xs:sequence> <xs:element name="Agent" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="ID" type="xs:ID" use="required"/> <xs:attribute name="IsCurrent-Owner" type="xs:boolean" use="required"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </pre>


```

<xs:attribute name="IsOrignator" type="xs:boolean"/>
<xs:attribute name="Agent-Type" use="optional"/>
<xs:attribute name="Agency-Number"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

element **Agent-Message/Sender-List/Agent**

diagram						
type	extension of xs:string					
properties	isRef	0	minOcc	1	maxOcc	unbounded
	content	complex				
attributes	Name	Type	Use	Default	Fixed	annotation
	ID	xs:ID	required			
	IsCurrent-Owner	xs:boolean	required			
	IsOrignator	xs:boolean				
	Agent-Type		optional			
	Agency-Number					
source	<pre> <xs:element name="Agent" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="ID" type="xs:ID" use="required"/> <xs:attribute name="IsCurrent-Owner" type="xs:boolean" use="required"/> <xs:attribute name="IsOrignator" type="xs:boolean"/> <xs:attribute name="Agent-Type" use="optional"/> <xs:attribute name="Agency-Number"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </pre>					

attribute **Agent-Message/Sender-List/Agent/@ID**

type	xs:ID
properties	isRef 0 use required
source	<code><xs:attribute name="ID" type="xs:ID" use="required"/></code>

attribute **Agent-Message/Sender-List/Agent/@IsCurrent-Owner**

type	xs:boolean
properties	isRef 0 use required
source	<code><xs:attribute name="IsCurrent-Owner" type="xs:boolean" use="required"/></code>

attribute **Agent-Message/Sender-List/Agent/@IsOriginator**

type	xs:boolean
properties	isRef 0
source	<code><xs:attribute name="IsOriginator" type="xs:boolean"/></code>

attribute **Agent-Message/Sender-List/Agent/@Agent-Type**

properties	isRef 0 use optional
source	<code><xs:attribute name="Agent-Type" use="optional"/></code>

attribute **Agent-Message/Sender-List/Agent/@Agency-Number**

properties	isRef 0
source	<code><xs:attribute name="Agency-Number"/></code>

element **Agent-Message/Recipient-List**

diagram	<pre> sequenceDiagram participant RL as Recipient-List participant Seq as ... participant AG as Agent RL --> Seq Seq --> AG Note over AG: 1..∞ </pre>
properties	isRef 0 content complex
children	Agent
source	<pre> <xs:element name="Recipient-List"> <xs:complexType> <xs:sequence> <xs:element name="Agent" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> </pre>

```

<xs:attribute name="ID" type="xs:ID" use="required"/>
<xs:attribute name="IsMandatory" type="xs:string" use="optional"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

element **Agent-Message/Recipient-List/Agent**

diagram						
type	extension of xs:string					
properties	isRef	0	minOcc	1	maxOcc	unbounded
	content	complex				
attributes	Name	Type	Use	Default	Fixed	annotation
	ID	xs:ID	required			
	IsMandatory	xs:string	optional			
source	<pre> <xs:element name="Agent" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="ID" type="xs:ID" use="required"/> <xs:attribute name="IsMandatory" type="xs:string" use="optional"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </pre>					

attribute **Agent-Message/Recipient-List/Agent/@ID**

type	xs:ID
properties	isRef 0 use required
source	<pre><xs:attribute name="ID" type="xs:ID" use="required"/></pre>

attribute **Agent-Message/Recipient-List/Agent/@IsMandatory**

type	xs:string
properties	isRef 0 use optional

source	<code><xs:attribute name="IsMandatory" type="xs:string" use="optional"/></code>
--------	---

element Agent-Message/Semantic-Content

diagram													
properties	<table><tr><td>isRef</td><td>0</td></tr><tr><td>minOcc</td><td>0</td></tr><tr><td>maxOcc</td><td>1</td></tr><tr><td>content</td><td>complex</td></tr><tr><td>mixed</td><td>false</td></tr></table>	isRef	0	minOcc	0	maxOcc	1	content	complex	mixed	false		
isRef	0												
minOcc	0												
maxOcc	1												
content	complex												
mixed	false												
children	Ontologies-List Context-Expression-Value												
attributes	<table><tr><th>Name</th><th>Type</th><th>Use</th><th>Default</th><th>Fixed</th><th>annotation</th></tr><tr><td>Use-Ontology</td><td>xs:string</td><td>optional</td><td></td><td></td><td></td></tr></table>	Name	Type	Use	Default	Fixed	annotation	Use-Ontology	xs:string	optional			
Name	Type	Use	Default	Fixed	annotation								
Use-Ontology	xs:string	optional											
source	<pre><xs:element name="Semantic-Content" minOccurs="0"> <xs:complexType mixed="false"> <xs:sequence> <xs:element name="Ontologies-List" minOccurs="0"> <xs:complexType> <xs:sequence> <xs:element name="Ontology" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> <xs:element name="Context-Expression-Value" minOccurs="0"> <xs:complexType mixed="true"> <xs:all minOccurs="0"/> </xs:complexType> </xs:element> <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> </xs:complexType> </xs:element></pre>												

	<pre> </xs:sequence> <xs:attribute name="Use-Ontology" type="xs:string" use="optional"/> <xs:anyAttribute namespace="##any"/> </xs:complexType> </xs:element> </pre>
--	--

attribute **Agent-Message/Semantic-Content/@Use-Ontology**

type	xs:string
properties	isRef 0 use optional
source	<pre><xs:attribute name="Use-Ontology" type="xs:string" use="optional"/></pre>

element **Agent-Message/Semantic-Content/Ontologies-List**

diagram	
properties	isRef 0 minOcc 0 maxOcc 1 content complex
children	Ontology
source	<pre> <xs:element name="Ontologies-List" minOccurs="0"> <xs:complexType> <xs:sequence> <xs:element name="Ontology" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </pre>

element **Agent-Message/Semantic-Content/Ontologies-List/Ontology**

diagram	
type	extension of xs:string
properties	isRef 0 minOcc 1 maxOcc unbounded

	content	complex				
attributes	Name	Type	Use	Default	Fixed	annotation
	Priority-Order	xs:unsignedByte	optional			
source	<pre> <xs:element name="Ontology" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </pre>					

attribute Agent-Message/Semantic-Content/Ontologies-List/Ontology/@Priority-Order

type	xs:unsignedByte
properties	isRef 0 use optional
source	<pre> <xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/> </pre>

element Agent-Message/Semantic-Content/Context-Expression-Value

diagram	
properties	isRef 0 minOcc 0 maxOcc 1 content complex mixed true
source	<pre> <xs:element name="Context-Expression-Value" minOccurs="0"> <xs:complexType mixed="true"> <xs:all minOccurs="0"/> </xs:complexType> </xs:element> </pre>

element Agent-Message/Conditions

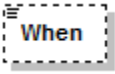
diagram	
properties	isRef 0 minOcc 0 maxOcc 1 content complex
children	When If
source	<pre> <xs:element name="Conditions" minOccurs="0"> <xs:complexType> <xs:choice> </pre>

```


<xs:element name="When" type="xs:string" minOccurs="0"/>
<xs:element name="If" type="xs:string" minOccurs="0"/>
</xs:choice>
</xs:complexType>
</xs:element>

```

element **Agent-Message/Conditions/When**

diagram	
type	xs:string
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<code><xs:element name="When" type="xs:string" minOccurs="0"/></code>

element **Agent-Message/Conditions/If**

diagram	
type	xs:string
properties	isRef 0 minOcc 0 maxOcc 1 content simple
source	<code><xs:element name="If" type="xs:string" minOccurs="0"/></code>

Appendix N: Schema XSD – Semantic Content Structure

element Semantic-Content

diagram													
properties	<table><tr><td>content</td><td>complex</td></tr><tr><td>mixed</td><td>false</td></tr></table>	content	complex	mixed	false								
content	complex												
mixed	false												
children	Ontologies-List Context-Expression-Value Rules												
attributes	<table><tr><th>Name</th><th>Type</th><th>Use</th><th>Default</th><th>Fixed</th><th>annotation</th></tr><tr><td>Use-Ontology</td><td>xs:string</td><td>optional</td><td></td><td></td><td></td></tr></table>	Name	Type	Use	Default	Fixed	annotation	Use-Ontology	xs:string	optional			
Name	Type	Use	Default	Fixed	annotation								
Use-Ontology	xs:string	optional											
source	<pre><xs:element name="Semantic-Content"> <xs:complexType mixed="false"> <xs:sequence> <xs:element name="Ontologies-List"> <xs:complexType> <xs:sequence> <xs:element name="Ontology" minOccurs="0" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/> <xs:attribute name="URI" type="xs:anyURI"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element></pre>												


```

</xs:element>
<xs:element name="Context-Expression-Value">
  <xs:complexType mixed="true">
    <xs:all/>
  </xs:complexType>
</xs:element>
<xs:any namespace="##other" maxOccurs="unbounded"/>
<xs:element name="Rules" minOccurs="0">
  <xs:complexType>
    <xs:all/>
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="Use-Ontology" type="xs:string" use="optional"/>
<xs:anyAttribute namespace="##any"/>
</xs:complexType>
</xs:element>

```

attribute Semantic-Content/@Use-Ontology

type	xs:string
properties	isRef 0 use optional
source	<xs:attribute name="Use-Ontology" type="xs:string" use="optional"/>

element Semantic-Content/Ontologies-List

diagram	
properties	isRef 0 content complex
children	Ontology
source	<pre> <xs:element name="Ontologies-List"> <xs:complexType> <xs:sequence> <xs:element name="Ontology" minOccurs="0" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/> <xs:attribute name="URI" type="xs:anyURI"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </pre>

element **Semantic-Content/Ontologies-List/Ontology**

diagram						
type	extension of xs:string					
properties	isRef	0	minOcc	0	maxOcc	unbounded
	content	complex				
attributes	Name	Type	Use	Default	Fixed	annotation
	Priority-Order	xs:unsignedByte	optional			
	URI	xs:anyURI				
source	<pre> <xs:element name="Ontology" minOccurs="0" maxOccurs="unbounded"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/> <xs:attribute name="URI" type="xs:anyURI"/> </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element> </pre>					


attribute **Semantic-Content/Ontologies-List/Ontology/@Priority-Order**

type	xs:unsignedByte
properties	isRef 0 use optional
source	<pre><xs:attribute name="Priority-Order" type="xs:unsignedByte" use="optional"/></pre>

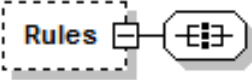
attribute **Semantic-Content/Ontologies-List/Ontology/@URI**

type	xs:anyURI
properties	isRef 0
source	<pre><xs:attribute name="URI" type="xs:anyURI"/></pre>

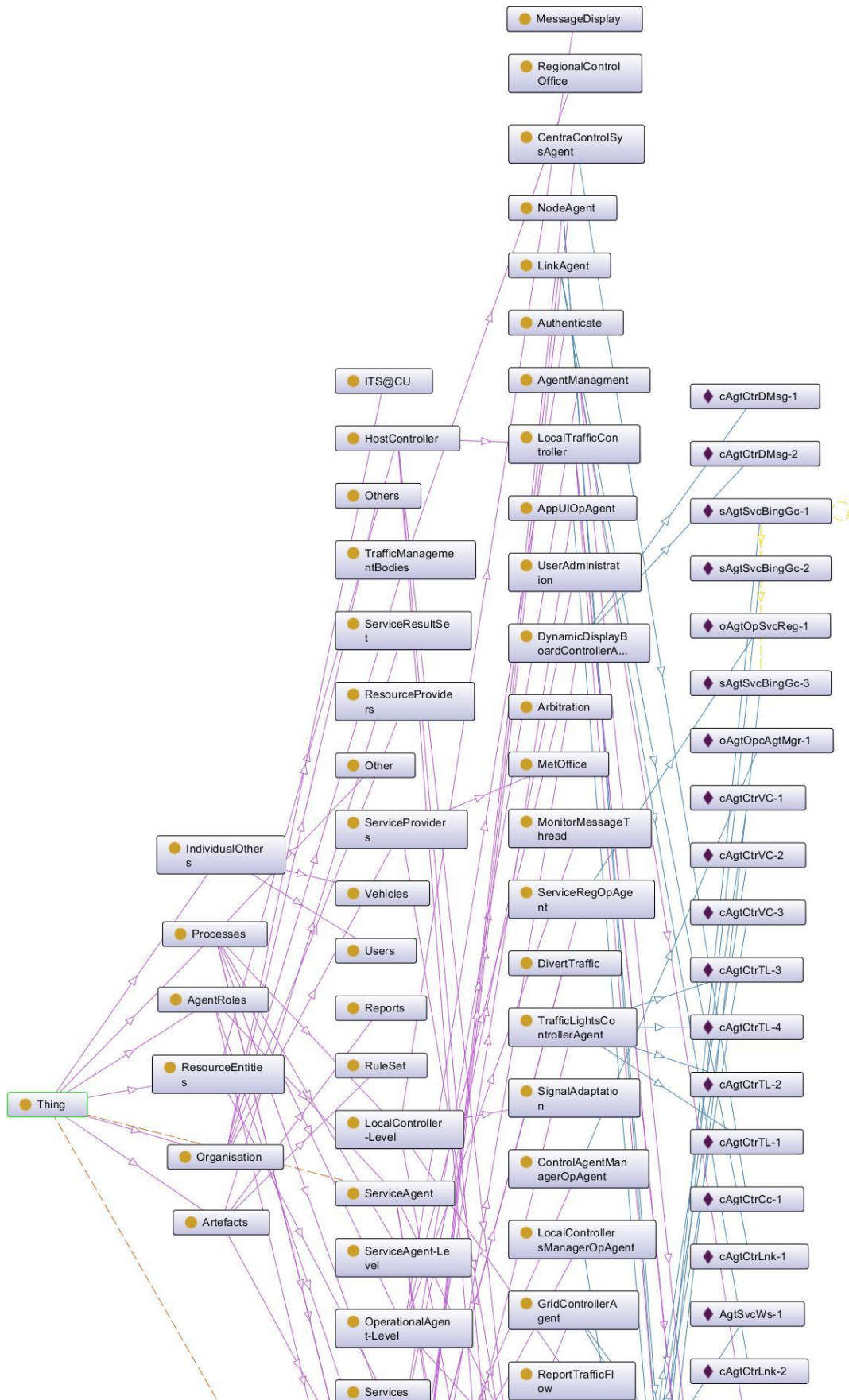
element **Semantic-Content/Context-Expression-Value**

diagram	 The diagram shows a rectangular box labeled "Context-Expression-Value" with a small square icon to its right, which is connected to a larger octagonal icon containing a grid pattern.
properties	isRef 0 content complex mixed true
source	<pre><xs:element name="Context-Expression-Value"> <xs:complexType mixed="true"> <xs:all/> </xs:complexType> </xs:element></pre>

element **Semantic-Content/Rules**

diagram	 The diagram shows a dashed rectangular box labeled "Rules" with a small square icon to its right, which is connected to a larger octagonal icon containing a grid pattern.
properties	isRef 0 minOcc 0 maxOcc 1 content complex
source	<pre><xs:element name="Rules" minOccurs="0"> <xs:complexType> <xs:all/> </xs:complexType> </xs:element></pre>

Appendix O: ITS@CU Ontology Structure



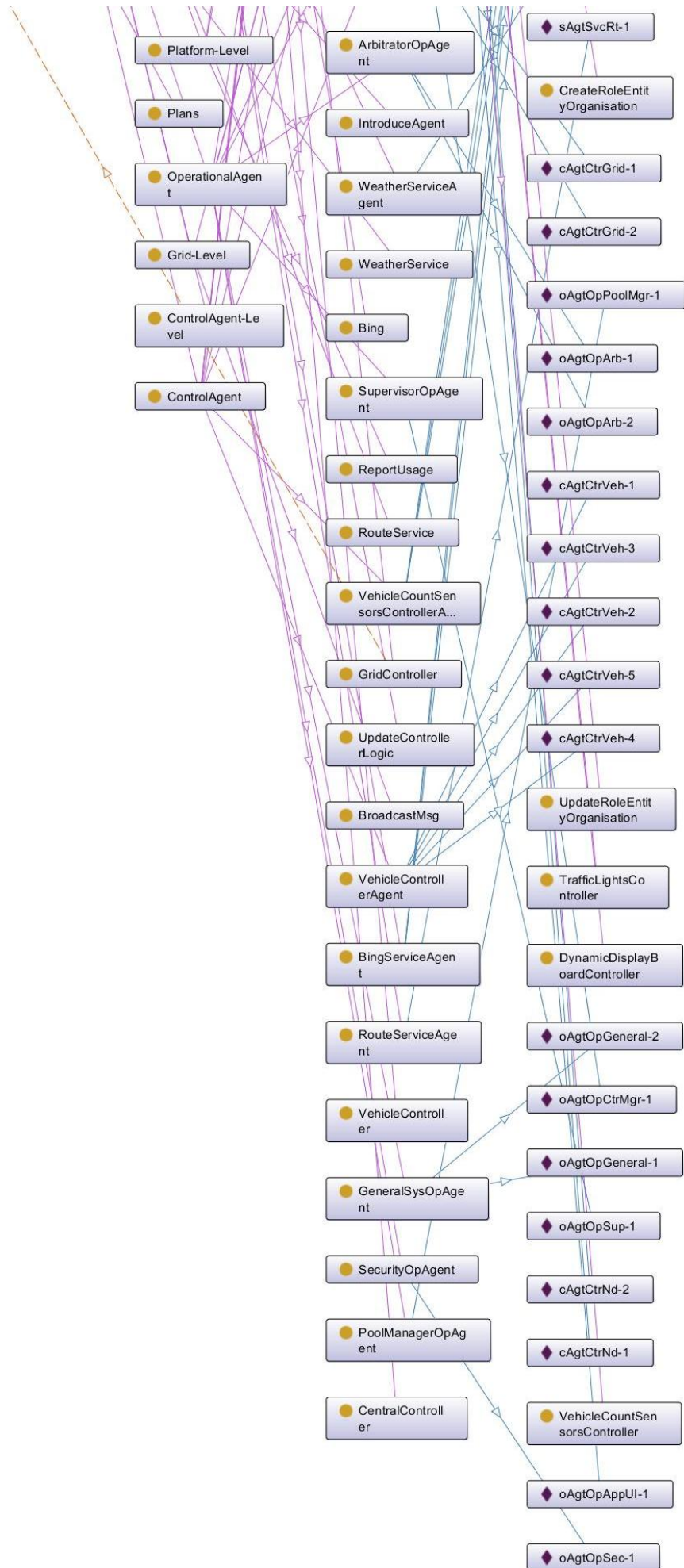


Figure 2: ITS@CU Management Application (Menu options)

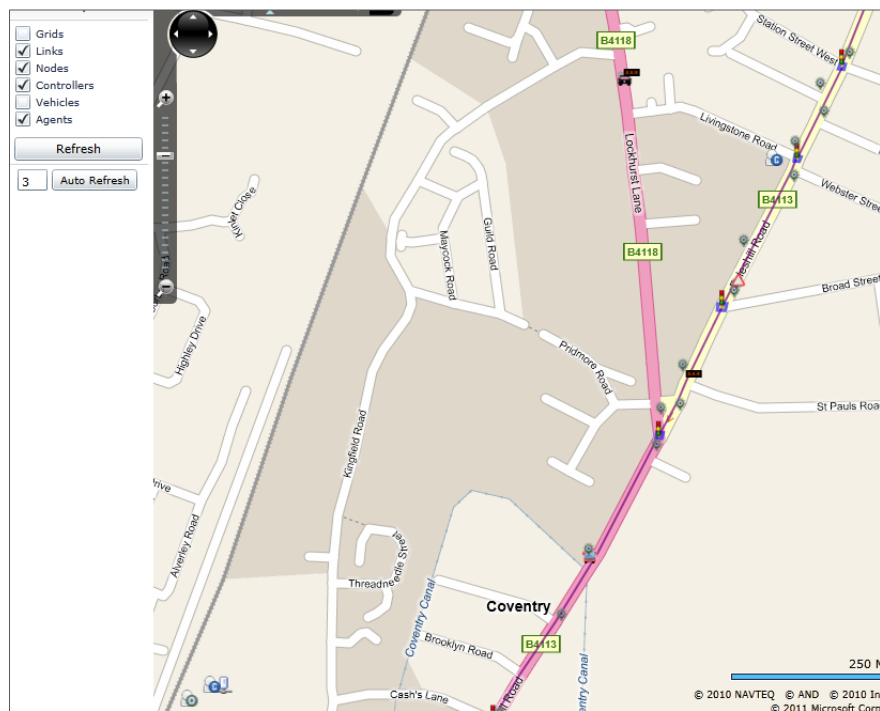


Figure 3: Monitoring view with Traffic Controls on a route

Adding new entities/components

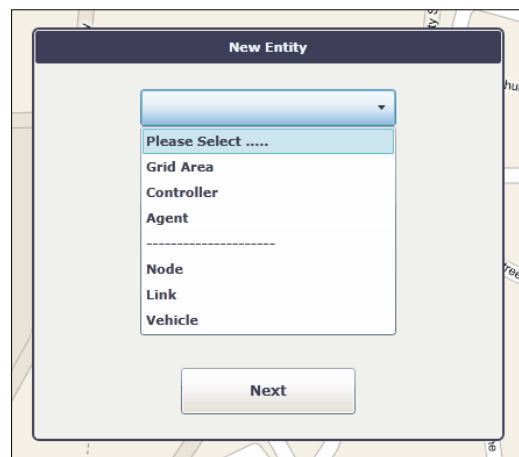


Figure 4: Adding new entities form

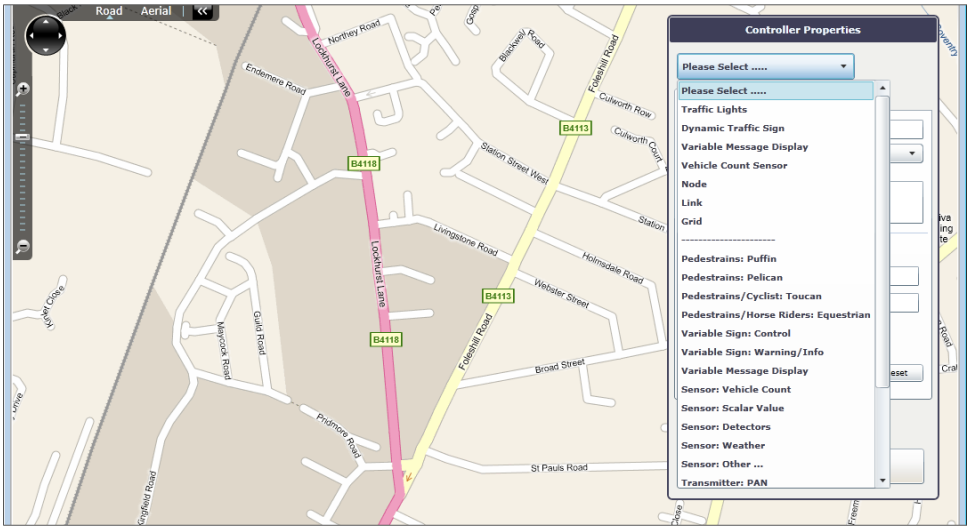


Figure 5: Adding a new Controller

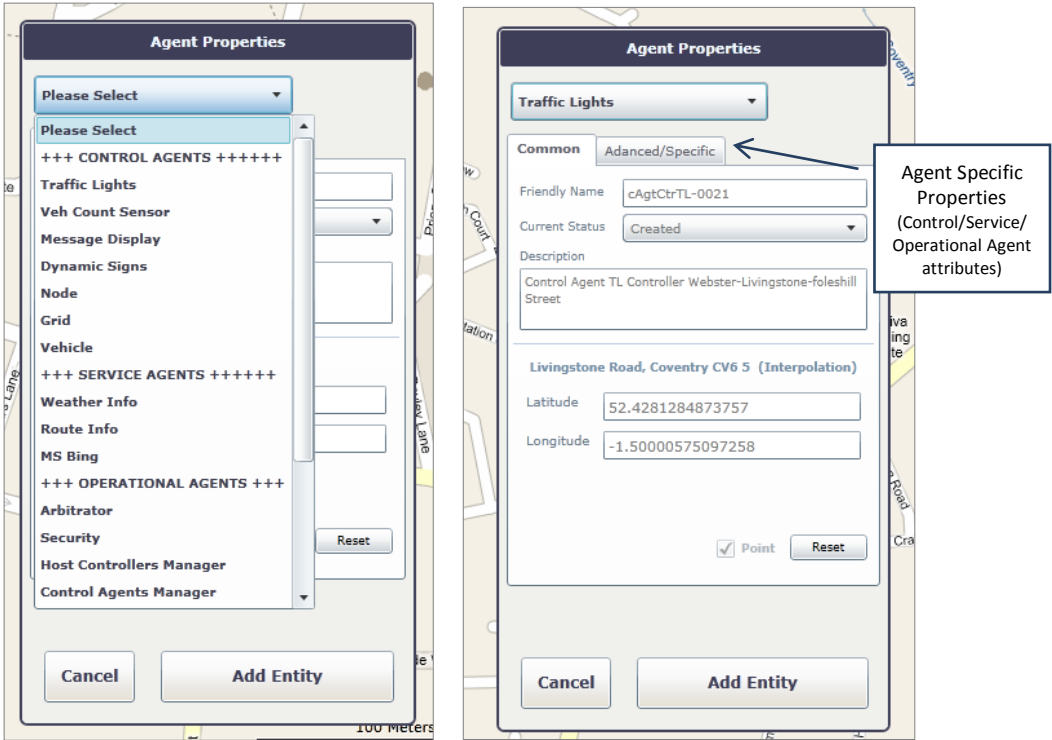


Figure 6: Adding a new Agent

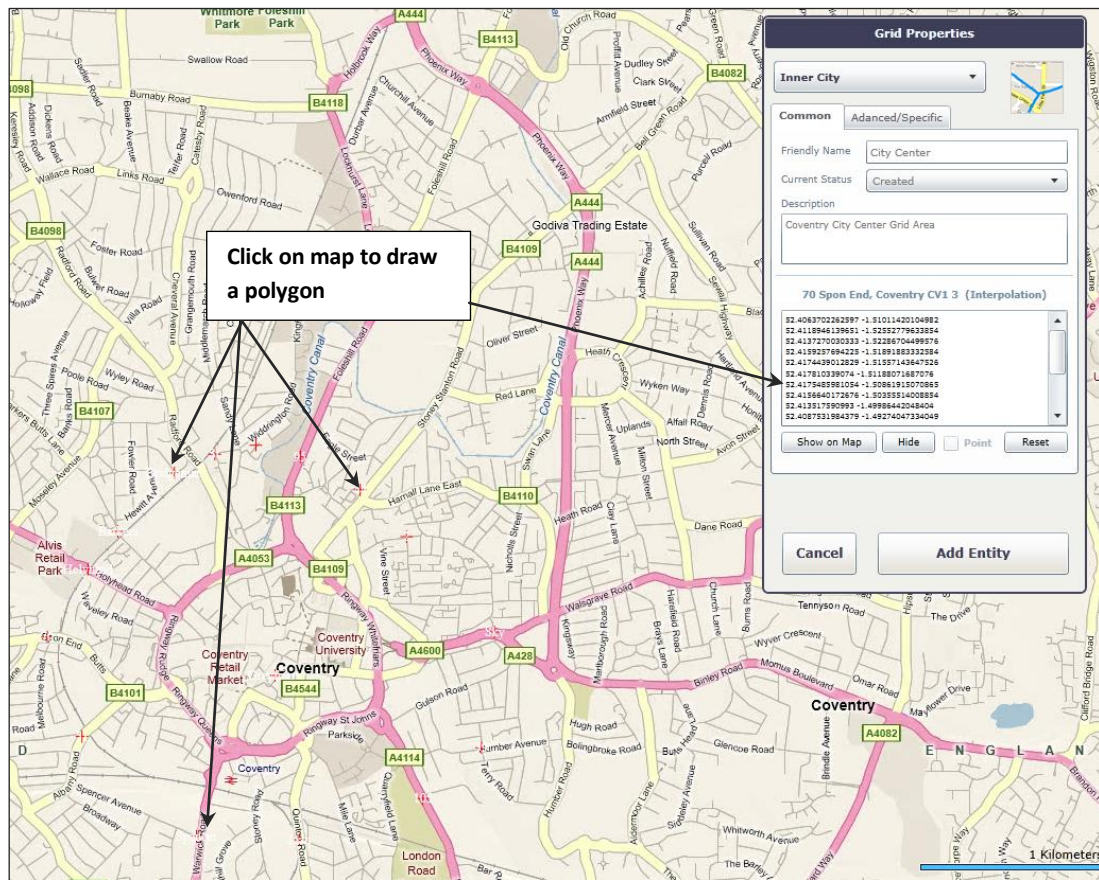


Figure 7: Add a new Grid Area

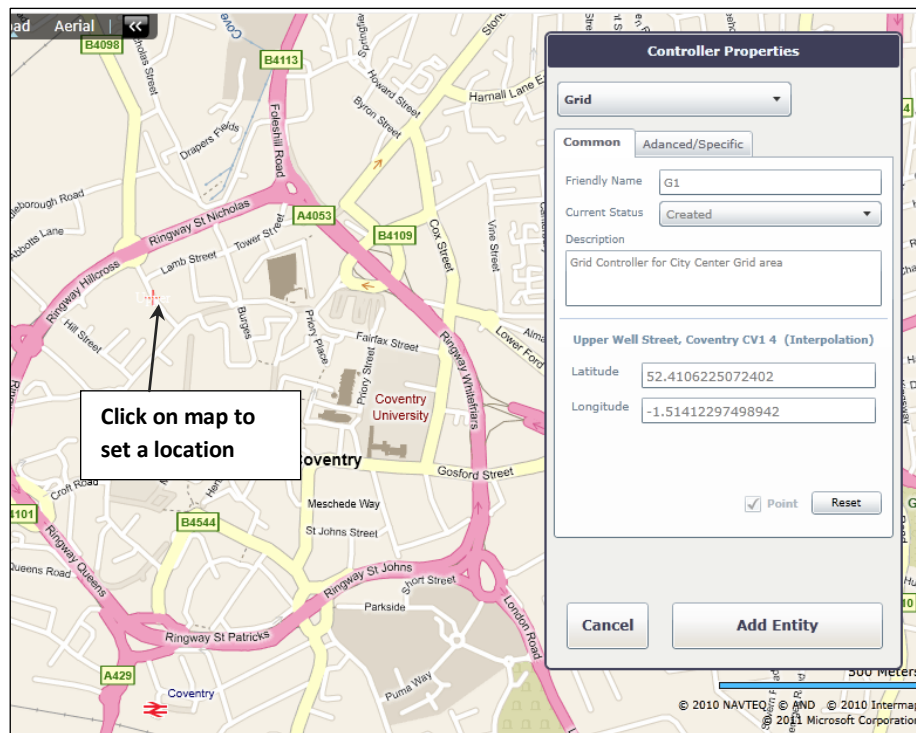


Figure 8: Add Grid Controller

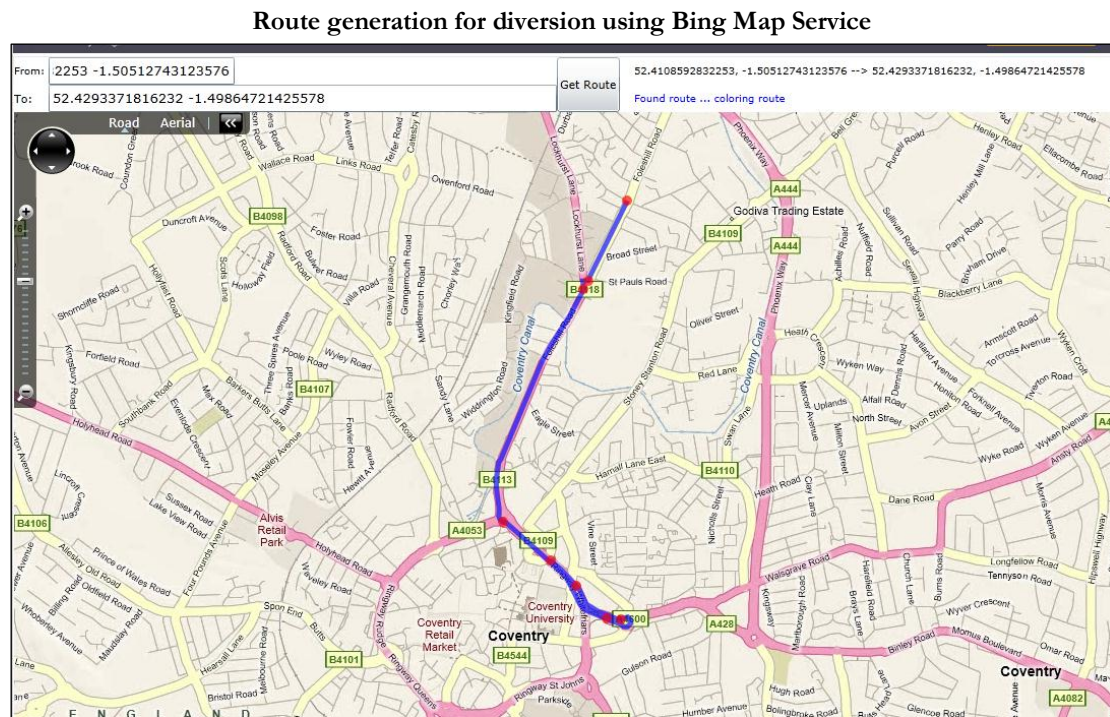


Figure 9: Setting route diversion for simulation purpose

2. ITS@CU Traffic Controls Management application (Grid Level)

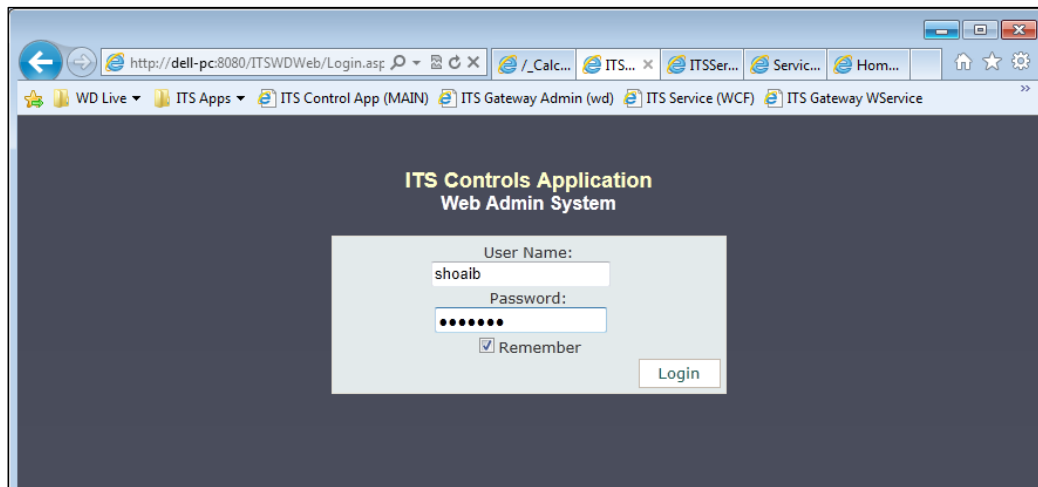


Figure 10: Traffic Controls Management (login Screen)

Recieve Time	User	Device Time	GPS Status	Battery	Device ID/IMEI
05/05/2011 15:05:38	NL02CJV	07:03:20	GPS/ON/Signal OK	0	
05/05/2011 15:05:20	NL02CJV	07:04:11	GPS/ON/Signal OK	0	
05/05/2011 15:05:20	NL02CJV	07:04:16	GPS/ON/Signal OK	0	
05/05/2011 15:03:35	NL02CJV	07:02:59	GPS/ON/Signal OK	0	
05/05/2011 15:03:30	NL02CJV	07:02:54	GPS/ON/Signal OK	0	
05/05/2011 15:03:25	NL02CJV	07:02:49	GPS/ON/Signal OK	0	
05/05/2011 15:03:20	NL02CJV	07:02:44	GPS/ON/Signal OK	0	
05/05/2011 15:03:15	NL02CJV	07:02:39	GPS/ON/Signal OK	0	
05/05/2011 15:03:10	NL02CJV	07:02:34	GPS/ON/Signal OK	0	
05/05/2011 15:03:05	NL02CJV	07:02:29	GPS/ON/Signal OK	0	
05/05/2011 15:02:59	NL02CJV	07:02:24	GPS/ON/Signal OK	0	
05/05/2011 15:02:55	NL02CJV	07:02:19	GPS/ON/Signal OK	0	

Figure 11: Current Controls Status monitoring

User	Last Login Time	Last LogOut Time	App. Version
Select shoalb	04/05/2011 00:59:31		ITSControlVeh_1.0.0.0
Select NL02CJV	29/05/2011 23:53:09		ITSControlVeh_1.0.0.0
Select olivierCar			

Figure 12: Controls Connectivity monitoring chart

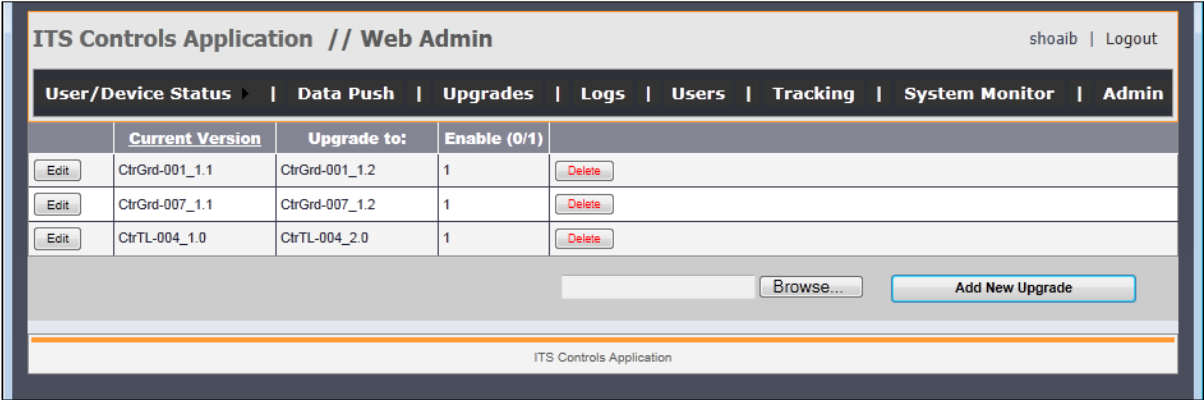


Figure 13: Controller updates (Default-Logic/Agent-Logic update)

3. ITS@CU Simulation Utility

	Agent ID	Friendly Name	Type	Status	Last Updated	Description	CreatedFirst		
Select	50001	cAgtCtrGrid-001	Grid	Created	10/09/2011 15:43:29	Control Agent for Grid Controller G1	10/09/2011 15:43:29	Edit	X
Select	50008	cAgtCtrGrid-002	Grid	Created	10/09/2011 17:49:50	Grid Control Agent gor G2	10/09/2011 17:49:50	Edit	X
Select	50009	oAgtArb-001	Arbitrator	Created	10/09/2011 17:53:00		10/09/2011 17:53:00	Edit	X

Figure 14: Setup simulation entities (Agent view)

ID	Type	Status	Owner	Enabled	Command
1025	Possibilities	Recieved	NL02CJV	<input type="checkbox"/>	<pre><!-- ACL type Messages, Read Agent XML doc --> <command> <action type="Vehicle Update" reg="NL02CJV" user-choice-allowed="true" semantics="veh" GUID="v10001"> <!-- ACL type Messages, Read Agent XML doc --> <command> <action type="Vehicle Update" reg="NL02CJV" user-choice-allowed="true" semantics="veh" GUID="v10001"></pre>
1026	MESSAGE	Recieved	shoalb	<input checked="" type="checkbox"/>	

Figure 15: Generate an Agent Message (behaviour testing/simulation)

Edit Commands

Command ID: 1025

Command Type: Possibilities

Current Status: Recieved

Owner: NL02CJV

Enabled: ☐

Command:

```
<!-- ACL type Messages, Read Agent XML doc -->
<command>
  <action type="Vehicle Update" reg="NL02CJV" user-choice-allowed="true" semantics="veh" GUID="v10001">
    <!-- ACL type Messages, Read Agent XML doc -->
    <command>
      <action type="Vehicle Update" reg="NL02CJV" user-choice-allowed="true" semantics="veh" GUID="v10001">
```

Update Cancel

Figure 16: Generate Agent Commands/Ontologies/Rules

4. Controller Applications

Using the MADF (described in *chapter 7, section 7.4*), A Controller simulation application was developed to simulate various traffic Controllers for example traffic lights Controllers, Variable message displays Controller, sensors Controller and dynamic traffic signs Controller.



Figure 17: Controller Setup Screen – Select and Configure a traffic Controller type



Figure 18: Controller Setup/Configuration Screen (Traffic Lights Controller Example)

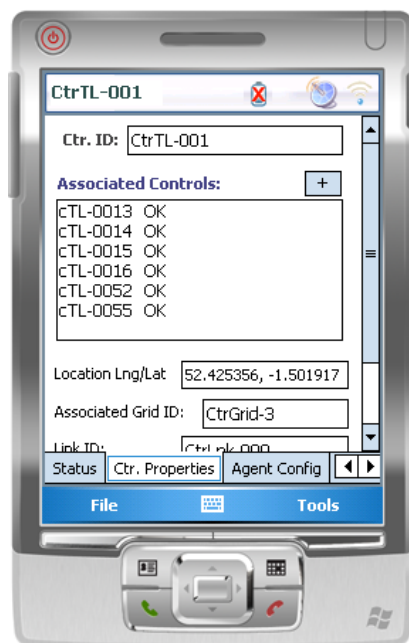


Figure 19: Controller properties Configuration (Traffic Lights Controller Example)

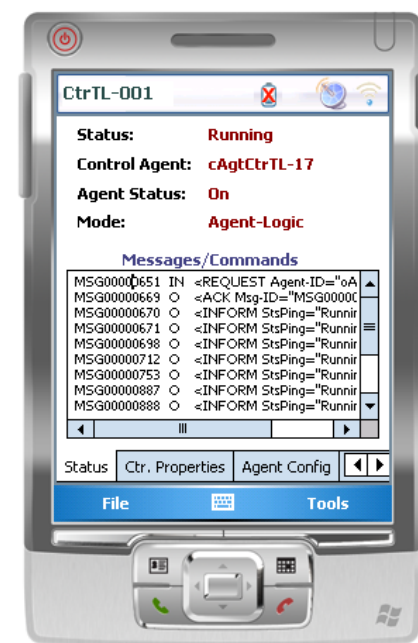


Figure 20: Controller Status Monitoring (Traffic Lights Controller Example)

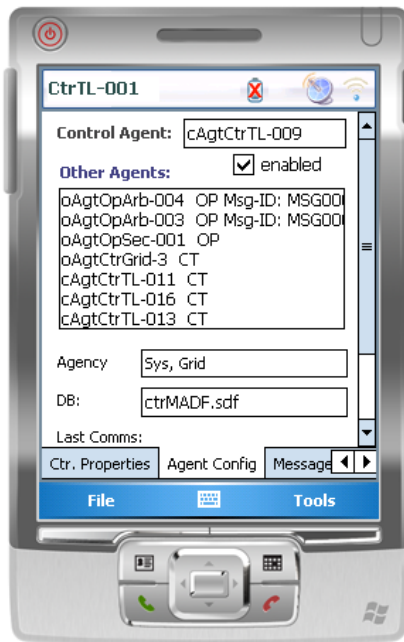


Figure 21: Hosted Control Agent Configuration and current Agents interaction status (Traffic Lights Controller Example)

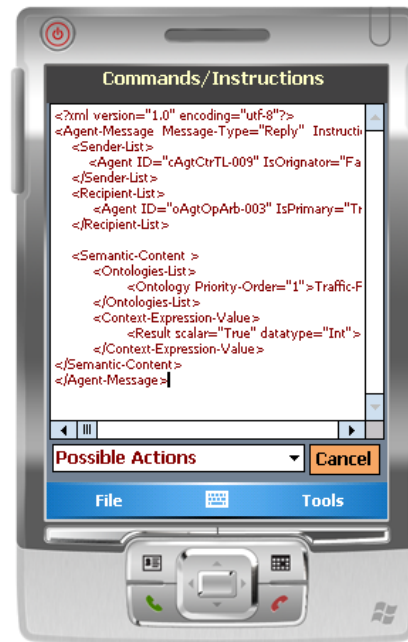


Figure 22: Agent Message composed by the hosted Control Agent on the Traffic Lights Controller application destined to an Operational Agent

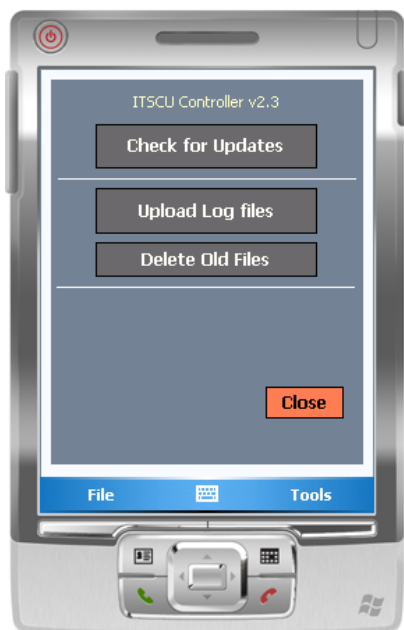


Figure 23: Controller Application management – Check for Agent-Logic update and clean local files

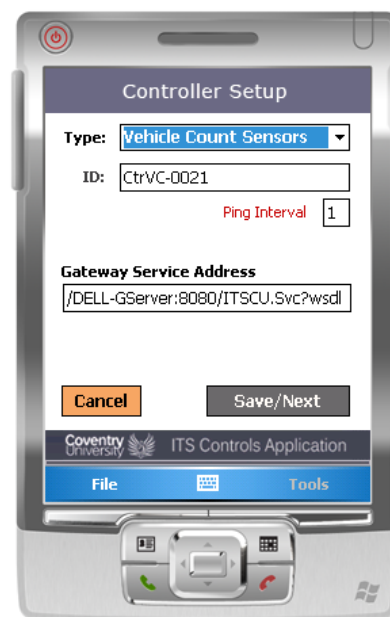


Figure 24: Controller Setup/Configuration Screen (Vehicle Count Sensor Controller Example)

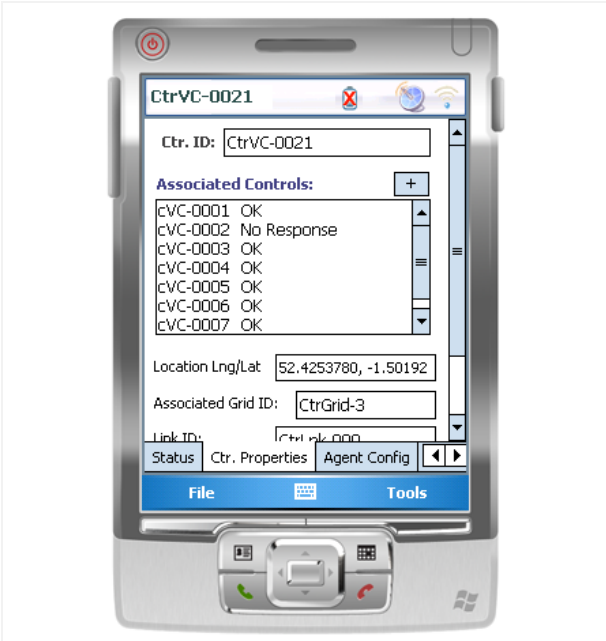


Figure 25: Controller properties Configuration
(Vehicle Count Sensor Controller Example)



Figure 26: Controller Status Monitoring
(Vehicle Count Sensor Controller Example)

5. Traffic Controls simulation Application

This application was develop using the MADF (described in chapter 7, *section 7.4*) in order to simulate the behaviour of various traffic Controls such as traffic lights, variable message displays, road sensors and dynamic speed limit signs. The traffic Controls are associated with Controllers (described in *section 5* above) to function.

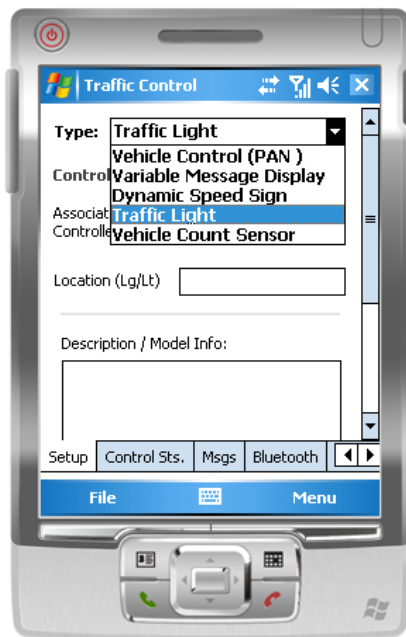


Figure 27: Traffic Control Setup Screen – Select and Configure a type

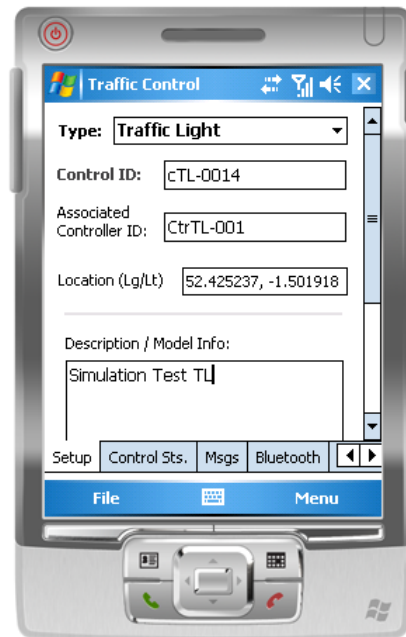


Figure 28: Control Setup/Configuration Screen (A Traffic Light Control Example associated with the Controller in *Figure 18*)



Figure 29: Control current Status and properties Configuration (Traffic Light Example)

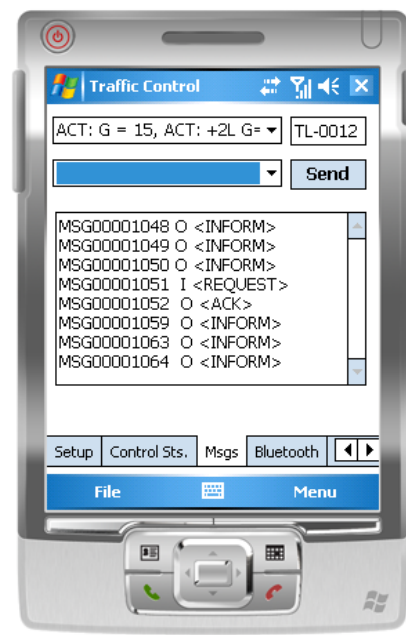


Figure 30: Commands view – for sending test commands to other Controls within a range

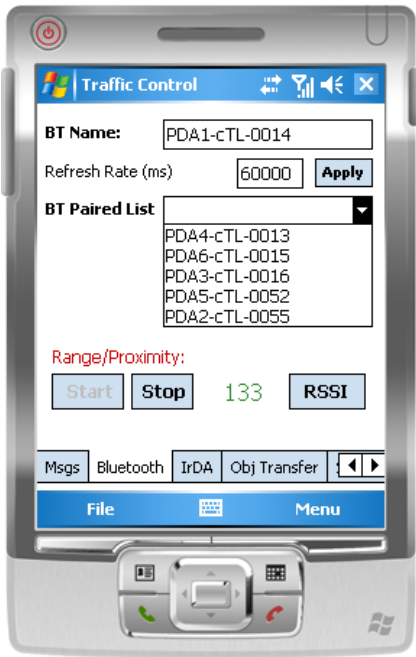


Figure 31: Bluetooth Connection configuration: pairing with other similar Controls within a range

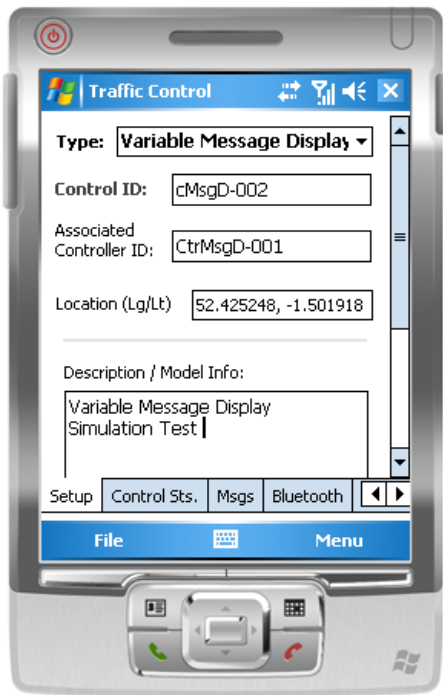


Figure 32: Control Setup/Configuration Screen (A Variable Message Display Example)

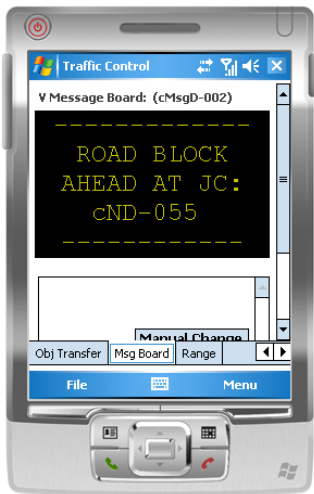


Figure 33: Control's current Status and properties Configuration (Variable Message Display Example)

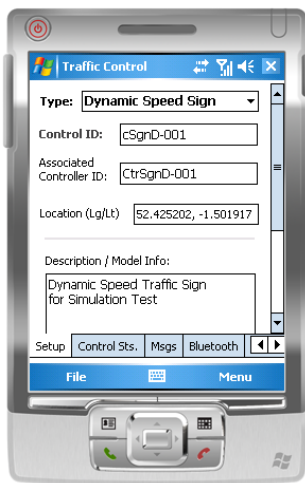


Figure 34: Control Setup/Configuration Screen (Dynamic Speed Sign Example)



Figure 35: Control's current Status and properties Configuration (Dynamic Speed Sign Example)

6. Vehicle Controller Application

Using the MADF (described in *chapter 7, section 7.4*), A Vehicle Controller application was developed to simulate on board vehicle Controller and its behaviour in the platform.

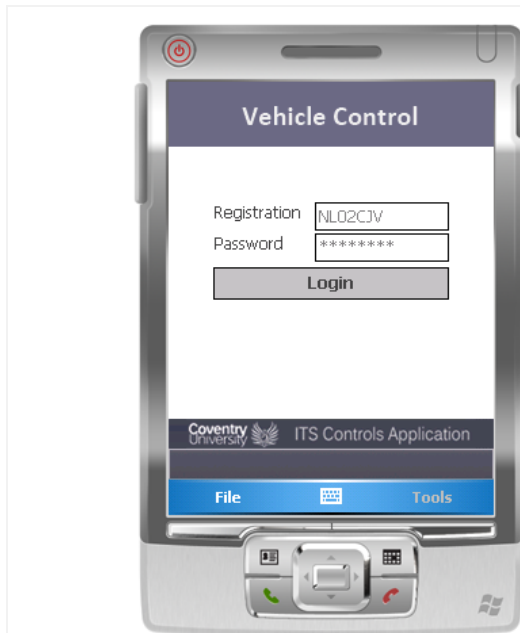


Figure 36: Vehicle Application Login Screen

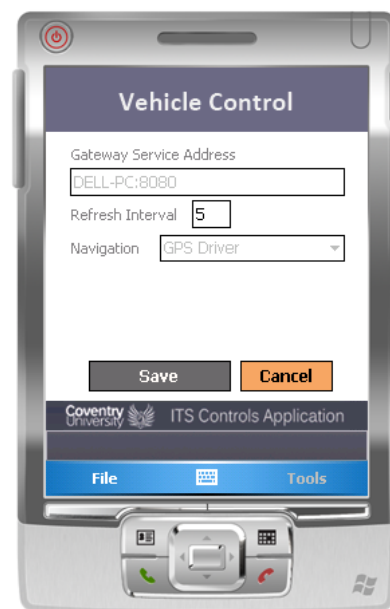


Figure 37: Setup/Configuration Screen

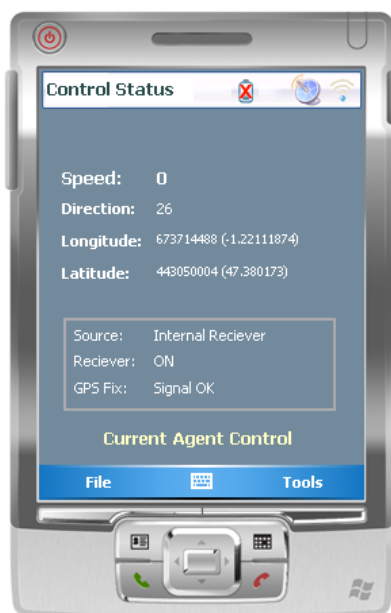


Figure 38: Vehicle status monitoring Screen

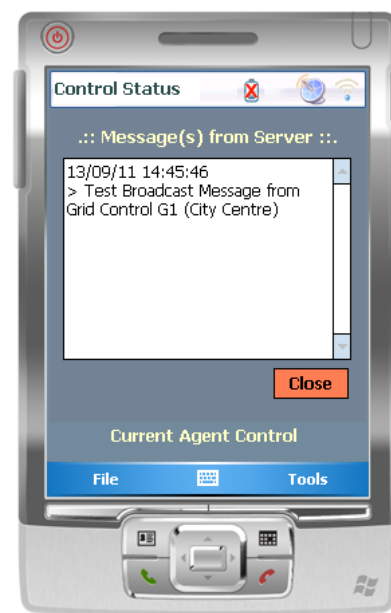


Figure 39: A broadcast Message from a Grid Controller



Figure 40: Notification Screen (from an Operational Agent to inform road block ahead)



Figure 41: In-App Navigation (TomTom 5) Software for automated Re-Routing based on the diversion route planned by Agents

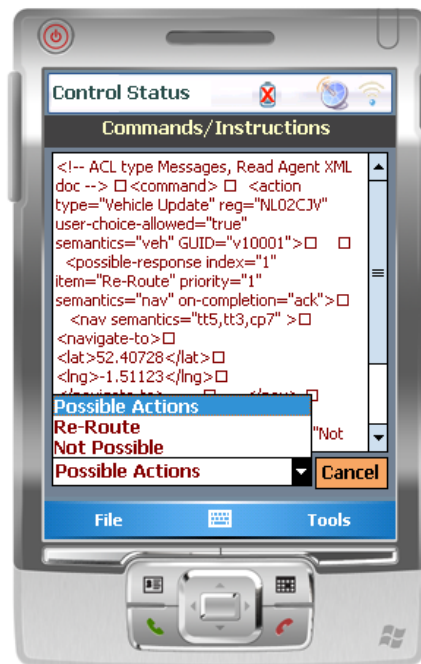


Figure 42: New Command Message from Grid Control Agent (Simulation view)