Cooperative Vehicle-Infrastructure Systems

D.CVIS.3.4	Final Architecture and System Specifications

SubProject No.	**1.2**	**SubProject Title**	**CAG**
Workpackage No.	**WP3**	**Workpackage Title**	**Architecture and system specifications**
Task No.	**n.a.**	**Task Title**	**D3.4**
Author(s)		**Arnor Solberg (QFREE), Andreas Schmid (PTV), Mick Baggen(Technolution), Francesco Alesiani (Mizar), Hannes Stratil (Efkon), Richard Bossom (Siemens), Silke Forkert (PTV), Marius Schlingelhof (DLR), Axel Burkert (PTV), Sjoerd Haverkamp (PEEK), Paul Mathias (Siemens), Zeljko Jeftic (ERTICO), Knut Evensen(Q-Free), Erik Olsen (Q-Free), Hans-Joachim Fischer (Q-Free), Imre Fazekas (Ygomi), Jean-Francois Gaillet (Ygomi)**	
Dissemination level PU/PP/RE		**PU**	
File Name		**DEL_CVIS_3.4_Final_Architecture_and_System_Specifications_v1.0.doc**	
Due date		**28 February 2010**	
Delivery date		**04 June 2010**	
Abstract	This document presents the final overall Architecture and System Specifications of the CVIS integrated project. It presents a view into CVIS core technologies architecture and reference applications.		

	Project supported by European Union DG INFSO IST-2004-2.4.12 eSafety -– Cooperative systems for road transport
Project reference	FP6-2004-IST-4-027293-IP
IP Manager	Paul Kompfner, ERTICO -– ITS Europe Tel: +32 2 400 0700, E-mail: cvis@mail.ertico.com

Control sheet

Version history			
Version	**Date**	**Main author**	**Summary of changes**
0.5	11-03-2010	Hans-Joachim Fischer	Update of D.CVIS:3.3
0.7	30-03-2010	Matthias Mann	Update of chapter 4.6 – Cooperative traffic information
0.9	20-05-2010	Imre Fazekas, Jean-Francois Gailletr	Update of security part
1.0	04-06-2010	Hans-Joachim Fisher / Erik Olsen	Final version

	Name	**Date**
Prepared	Erik Olsen / Hans-Joachim Fischer	09-03-2010
Reviewed	Knut Evensen, Matthias Mann, Jean-Francois Gaillet	26-04-2010
Authorized	Paul Kompfner	04/06/2010

Circulation	
Recipient	**Date of submission**
European Commission	04/06/2010
Project Consortium	04/06/2010

Table of Contents

Executive Summary

This document has been created to give a complete overview of all building blocks in the CVIS architecture in one place. It provides the integrated CVIS architecture specification to set up cooperative systems based on the architecture specifications developed in the different CVIS sub-projects.

CVIS has also provided a core input to COMeSafety architecture which recently has become both an ISO and European ETSI standard. It is therefore clear that this deliverable, which has been updated with the latest results from CVIS and standardisation, is highly relevant. The results have already been transferred to a number of other projects. Large parts of this document has since become ETSI and ISO standards. The basic architecture and concepts innovated by CVIS and documented in this report, forms the basis for the global understanding of what Cooperative Systems are. CVIS has also provided support to the EU/US Task Force on Cooperative Systems based upon the architecture and results obtained by CVIS.

The document describes how the future of cooperative systems can look like. The emphasis on an integrated cooperative system consisting of applications, facilities, execution and communication infrastructure together with legacy systems, makes it a highly innovative design. For future deployment it is very important, that existing legacy systems can easily be connected or integrated in a cooperative system.

This document includes:

A high level introduction to the CVIS architecture.

A description of the basic common functionalities, which the project calls "Facilities". These are described in two parts:

- "Basic Facilities" comprise CVIS building blocks in the domain of "Information and Communication Technology" (ICT),
- "Domain Facilities" comprise CVIS building blocks in the domain of "Intelligent Transport Systems" (ITS).

D.CVIS.3.4 is based on and refers to other architecture specification documents developed in CVIS, in particular the D.SP3.2 architecture specification documents of the sub-projects. It constitutes a refinement of D.CVIS.3.3.

D.CVIS.3.3 is based on and refers to other architecture specification documents developed in CVIS, in particular:

The D.CVIS.3.2 "High Level Architecture" (HLA) and the D.CVIS.3.1 "Reference Architecture". These architecture specification documents focus on general architecture principles applied in CVIS.

The D.SP.3.1 architecture specification documents. There is one D.SP.3.1 architecture specification document for each of the CVIS sub-projects. These documents include specifications of facilities and applications provided by the respective sub-projects including their internal design.

To make this IP level deliverable self contained and to provide a complete "story" there are some overlaps between D.CVIS.3.4 and the architecture specification documents outlined

above. However, the emphasis of D.CVIS.3.4 is to provide the overall picture and to specify how the different parts of the CVIS system are related and how these parts collaborate to fulfil their obligations. Thus, it provides an integrated specification of deliverables provided in different sub-projects.

The relationships and differences between the current set of architecture specification documents are as follows:

The CVIS architecture principles, some overall architecture descriptions and some general archetypical scenarios are described in the HLA and the reference architecture specification documents. The content of these specification documents are mainly referred to in this document, however, some principles and archetypical scenarios are repeated here as to put specified features and scenarios in its context.

The D.SP.3.1 architecture specification documents focus on specifying facilities and applications provided by the different CVIS sub-projects, including their internal design. This specification uses these specification documents as baseline and focuses on how the different parts are put together in the CVIS overall architecture and how these collaborate. Thus, the focus of D.CVIS.3.3 and D.CVIS.3.4 is on the external interfaces of the different facilities and applications as well as how they interoperate. For the internal design of these facilities and applications we refer to the D.SP.3.1 documents.

Figure 1 below shows the set of developed and planned architecture specification documents in CVIS and it depicts their relationships. (The bold red arrow in the figure points to this document (D.CVIS.3.4)).

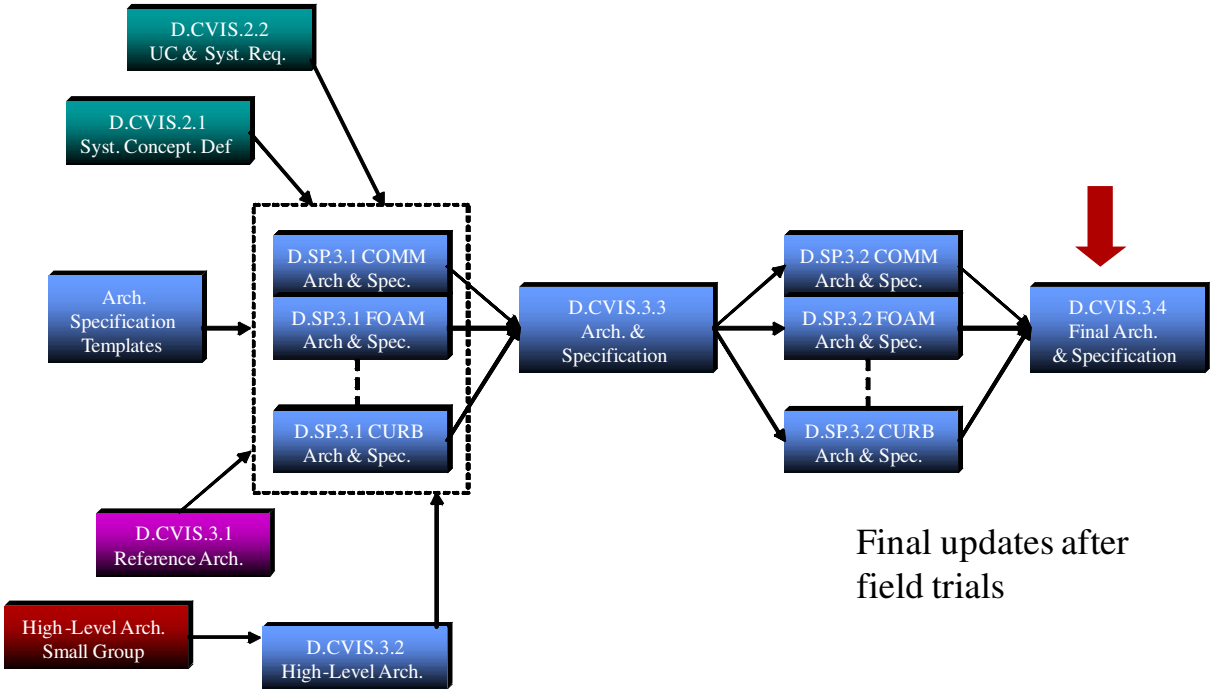

Figure 1: Architecture specification documents and their relationships

All the above depicted documents are "stand-alone" documents and can be read independently of the others. The content of the currently available architecture documents is briefly listed below:

"Architecture Specification Templates" is a family of documents providing guidelines and templates for harmonised architecture specifications in the CVIS project.

"D.CVIS.3.1 "Reference Architecture" describes general principles of the CVIS architecture

"D.CVIS.3.2 "High-Level Architecture" specifies the overall CVIS architecture.

"D.SP.3.2 "Final Architecture and System Specification" (one for each sub-project) specifies the applications and facilities provided by each of the sub projects in detail by the end of the CVIS project.

"D.3.4 "Final Architecture and System Specification" (this document) specifies the overall CVIS architecture and provides specifications of the services provided by each of the facilities and applications.

Abbreviations and definitions

Abbreviation	Definition
AA	Authentication and Authorization
2G	Second generation cellular phone technology, e.g. GSM. For ITS (ISO 21212)
3G	Third generation cellular phone technology, e.g. UMTS. For ITS (ISO 21213)
Access layer	Merged OSI layers 1and 2 as specified in the ITS station reference architecture
AIDE	Adaptive Integrated Driver-vehicle interfacE
API	Application Programming Interface
Application	Software bundle providing "End User Services"
AWT	Abstract Windowing Toolkit
BL	Bus Lane
Bundle	OSGi term denoting a software service packaged into a JAR file that can be deployed on the OSGi platform.
CAG	Core Architecture Group (horizontal CVIS subproject leading the technical work in the project)
CALM	Communication Access for Land Mobiles - this is the work title of a basic set of CEN/ISO communication standards for cooperative ITS
CAN	Controller Area Network
CDDF	CALM Device Driver Framework
CF&F	Cooperative Freight and Fleet applications. A CVIS sub-project
CINT	Cooperative Inter-Urban Applications. A CVIS sub-project
CME	CALM Management Entity. Part of the CALM ITS station and communication management. Term no more supported in latest version of CALM standards.
COMM	COMMunication & networking. A CVIS sub-project
COMO	COoperative MOnitoring. A CVIS sub-project
CTA	Co-operative Traveller Assistance
CURB	Cooperative URban Applications. A CVIS sub-project
CVIS	Cooperative Vehicle-Infrastructure Systems

Abbreviation	Definition
DATEX 2	DATEX standard was developed for information exchange between traffic management centres, traffic information centres and service providers and constitutes the reference for applications that have been developed in the last 10 years. The second generation DATEX 2 specification now also pushes the door wide open for all actors in the traffic and travel information sector. (http://www.datex2.eu/)
DB	Data Base
DDS	Distributed Directory Service
DEPN	DEPloyment eNablers
DG	Dangerous Goods
DG preferred network	Network where all road links are classified whether they can be used for DG transports or not
DM	Device Management
Driver	Person conducting a vehicle
DSRC	Dedicated Short Range Communications - ISO/CEN/ETSI standards. Backscatter technology at 5.8 GHz.
EDA	Enhanced Driver Awareness
EFCD	Enhanced Floating Car Data (same as XFCD)), created on occasion. Normally referring to data elaborated in the COMO process "Computation of Local Traffic State"
ETA	Estimated Time of Arrival
Facilities layer	Merged OSI layers 5, 6 and 7 as specified in the ITS station reference architecture
Facility	Software bundle providing services to be used by applications or other facilities
FCD	Floating Car Data
FOAM	Framework for Open Application Management. A CVIS sub-project
FRAME	Project name for European ITS framework architecture
Gateway	1. A device that allows to securely link the CAN network to the IP network. 2. Functionality of the ITS station (ISO 21217) / device to interconnect networks.
Ghost Driver	Driver conducting a vehicle contrary to the prescribed direction of traffic

Abbreviation	Definition
GNSS	Global Navigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications, 2G version of the cellular phone system.
GSP	Geo-Spatial Platform
GST	Global Systems for Telematics
GUI	Graphical User Interface
HLA	High Level Architecture
HMC	Host management centre
HMI	Human Machine Interface
ICT	Information and Communication Technology
IdP	Identity Provider
IME	Interface Management Entity. Part of the CALM ITS station and communication management. Term no more supported in latest version of CALM standards.
IN-SAP	SAP between access layer (I: Interface) and networking & transport layer (N: networking) specified in ISO 21217. Formerly referred to as C-SAP (ISO 21218).
IP	Integrated Project
IPv6	Internet Protocol version 6
IR	Infra Red (ISO 21214)
ITS	Intelligent Transport Systems
JAAS	Java Authentication and Authorization Service
JVM	Java Virtual Machine
LDM	Local Dynamic Map. Standardized data base containing geo-referenced data that is always available in vehicle and road-side sub-systems.
Legacy system	1. Existing system to which the CVIS platform is attached. These may be the existing system in a vehicle, e.g. the CAN bus or XFCD generation, in a RSU, e.g. the loop or traffic light controllers, or a centre, e.g. the computation of the centre-wide traffic state already existing in a TCC. 2. Communication equipment used in ITS, but according to existent non-ITS standards.

Abbreviation	Definition
LOS	1. Level Of Service 2. Line-of-sight (in communication)
LUTC	Local Urban Traffic Control
M5	CALM Microwave medium at 5 GHz, based on IEEE 802.11 (p) (ISO 21215)
MI-SAP	SAP between ITS station and communication management (M: Management) and access layer (I: Interface) specified in ISO 21217. Formerly referred to as M-SAP (ISO 21218).
NDM	Network Dynamic Map
Networking & transport layer	Merged OSI layers 3 and 4 as specified in the ITS station reference architecture
NME	Network Management Entity. Part of the CALM ITS station and communication management. Term no more supported in latest version of CALM standards.
OBU	On Board Unit
OEM	Original Equipment Manufacturer
OMA	Open Mobile Alliance
OSGi	Open Services Gateway initiative
OSI	Open Systems Interconnection; ISO-OSI layered model for communication protocols
PAP	Policy Administration Point
PDA	Personal Digital Assistant
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
POMA	POsitioning and MApping. A CVIS sub-project
PSAP	Public Safety Access Point
Q-API	LDM Query API used by CVIS applications to get access to COMO data
QoE	Quality of Environment
QoS	Quality of Service
RSSI	Received Signal Strength Indicator (used for range measurements)
RSU	Road-Side Unit
RTIG	Real Time Information Group

Abbreviation	Definition
SAML	Security Assertion Markup Language
SAP	Service Access Point; functional interface used in the ISO-OSI model
SC	Secure Communication
SCE	Secure Communication Engine
SOAP	Simple Object Access Protocol
SP	Sub-Project
SSO	Single Sign-On
TLC	Traffic Light Controller
TMC	Traffic Message Channel
TMC/NSP	Traffic management centre/ Navigation Service Provider
TPEG	Transport Protocol Experts Group
Traveller	A person planning or making a journey.
Tree information	Information provided by legacy systems concerning their sensors and actors
UC	Use Case
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunications System. 3G version of the cellular phone system.
URI	Uniform Resource Identifier
UTC	Universal Time Coordinated
V2I	Vehicle to Infrastructure (communication)
VMS	Virtual Message Sign
WiFi	Wireless Fidelity
WP	Work Package
XACML	eXtensible Access Control Markup Language
XFCD	eXtended Floating Car Data

Reference Documentation

Ref.	Short name	Document name	Date
RD01	CVIS Reference Architecture	DEL_CVIS_3.1_Reference_architecture_v1.2	2006-12-09
RD02	CVIS High Level Architecture	DEL_CVIS_3.2_High_Level_Architecture_v1.0	2007-03-02
RD03	D.CF&F.3.2	D.CF&F.3.2 Architecture and system specifications v1	2010-03-31
RD04	D.CINT.3.2	D.CINT.3.2 Architecture and system specifications v1	2010-06-04
RD05	D. COMM.3.2	D.COMM.3.2 Architecture and system specifications v1	2010-06-07
RD06	D.COMO.3.2	D.COMO.3.2 Architecture and system specifications v2.0	2010-06-03
RD07	D.CURB.3.2	D.CURB.3.2 Architecture and system specifications v1	2010-06-04
RD08	D.FOAM.3.2	D.FOAM.3.2 Architecture and system specifications v1.2	2010-02-18
RD09	D.POMA.3.2	D.POMA.3.2 Architecture and system specifications v1	2010-05-12

1 Introduction

D.CVIS.3.4 is the final IP level architecture deliverable of WP3 (see Figure 1). The rationale of this deliverable is to provide a coherent specification of the CVIS architecture. It provides the integrated CVIS architecture specification based on the architecture specifications developed in the different CVIS sub-projects.

Note that the final deployment architecture might be slightly different from the one developed and presented in CVIS. In particular, wherever safety and security aspects are concerned, e.g. road-side traffic control systems, traffic management centres, system responsibilities must be unambiguous. This will probably lead to closed sub-systems with interfaces that are physically different from the ones used in the CVIS project but in any case will make available the same data to potential applications.

All CVIS use cases and requirements can be found in D.CVIS.2.3 "Final System Requirements". The requirements will be used in the validation process but will not be included in this document in order to avoid overlap between deliverables and avoid possible conflicts between different versions of requirement sets.

1.1 Intended audience

The intended audience of this document are all stakeholders interested in the CVIS architecture specifications, in particular those who want to understand the overall CVIS architecture. Five main types of audiences can be distinguished:

1. System architects needing to understand their context and the overall CVIS system architecture.
2. System developers implementing various cooperative CVIS based functions.
3. The European Commission who is supporting the CVIS project.
4. Correlated projects in the area of cooperative systems.
5. External stakeholders who would like to understand the CVIS system architecture.

1.2 Document structure

This document is divided into three main parts:

Part I introduces the CVIS overall architecture, the main sub-systems and components, main information flows and domain concepts.

Part II provides specification of the CVIS facilities and infrastructures including the basic facilities, the domain facilities the communication infrastructure and the execution infrastructure.

Part III presents specifications of the CVIS applications.

PART I CVIS overall architecture

2 CVIS overview

2.1 Cooperative systems - setting the scene

CVIS developments are designed for a new way of cooperation and communication. It will create a break-through of today's ITS development and replace today's patterns of ITS systems. In the cooperative vision, vehicles, even all mobile traffic participants, road-side infrastructure and centre systems are no longer seen in a rather hierarchical relationship concerning communication and information processing. Instead all participants are seen as "nodes" in a common "network". In addition to this conceptual change, the physical wireless communication enhancements enables the "nodes" to be always connected to common networks and to communicate rather freely with each other according to their needs. This is often referred to as being the "always on" type of communications.

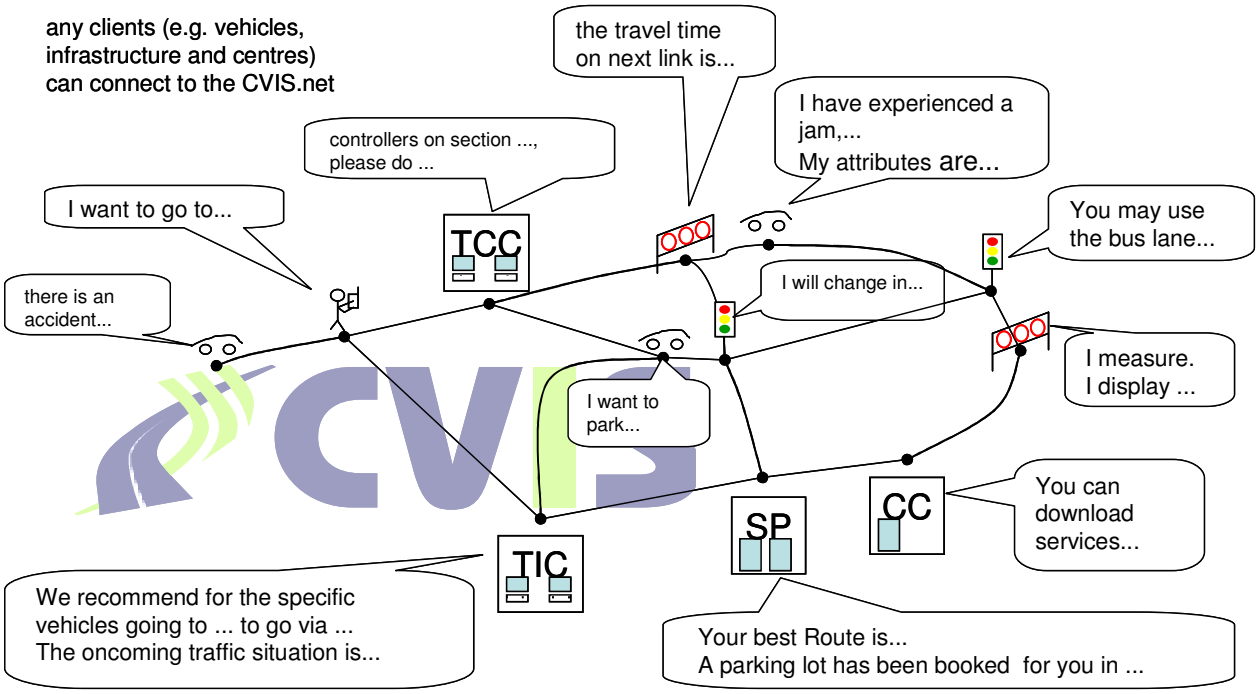

Figure 2: CVIS system overview

This new view of the relationships between systems provides a mighty capability to design various known ITS applications in a very different and more efficient manner and also opens up the door to the development of unprecedented new applications.

However the new view includes the challenge that the system remains manageable. Safety, security, privacy, stability of the system and its communication links must be achieved. The applications' needed changes of the over time must be managed and finally all technology must be usable in a way to respect the organisational structures of today and the future, as well as business needs.

CVIS contributes to this, providing:

- A communication solution, which is based on CALM specifications including IPv6.
- A middleware layer for managing the lifecycle and/or the deployment of application

software.

- A set of basic functional components, called "Facilities". These are split into two groups:
    - "Basic Facilities", supporting access of applications to necessary communication functions and to a security framework.
    - "Domain Facilities", supporting applications with a core ITS related tool set e.g. positioning, location referencing and getting traffic status information.
- A set of demonstration applications in urban, inter urban and in freight and fleet areas.

CVIS Top level architecture

The CVIS top level architecture is shown in Figure 3. The systems in the vehicle, at the road-side and in the central systems consist of:

- **A host computer** in the vehicle, at the road-side and in the central systems running cooperative CVIS applications. It encompasses a Java based middleware providing ITS services and facilities for easy development and life-cycle management of ITS applications.
- **A mobile router** in the vehicle and a similar access router at the road-side providing seamless communication facilities to the ITS applications based on the CALM standard.
- **A gateway** to the existing or legacy systems either at the road-side or in the vehicle.
- **A border router** at the road-side and in the central systems connecting to the Internet.

Also refer to chapter 2.2 on page 21 for further details.

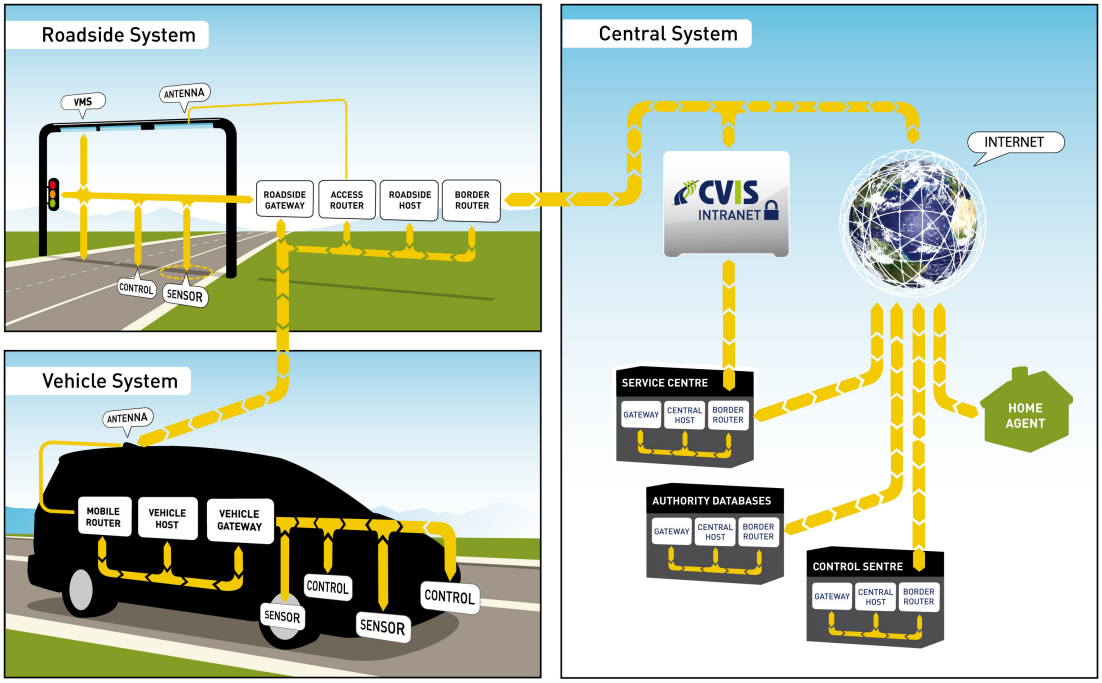

Figure 3: CVIS tope level architecture

In CVIS, cooperativeness is achieved through the basic set of CALM ITS communication standards presented in Figure 5, which enable seamless communications to cooperative applications as illustrated in Figure 4.

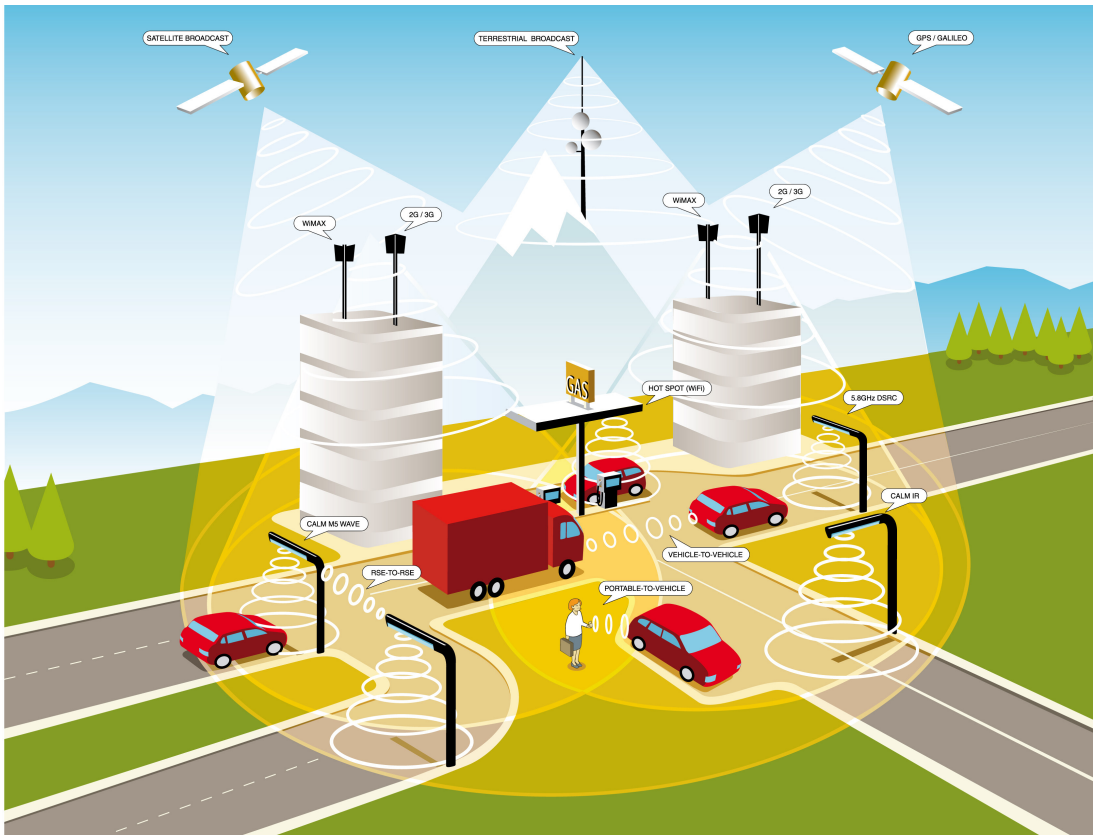

Figure 4: Continuous communication as a basis for cooperative systems

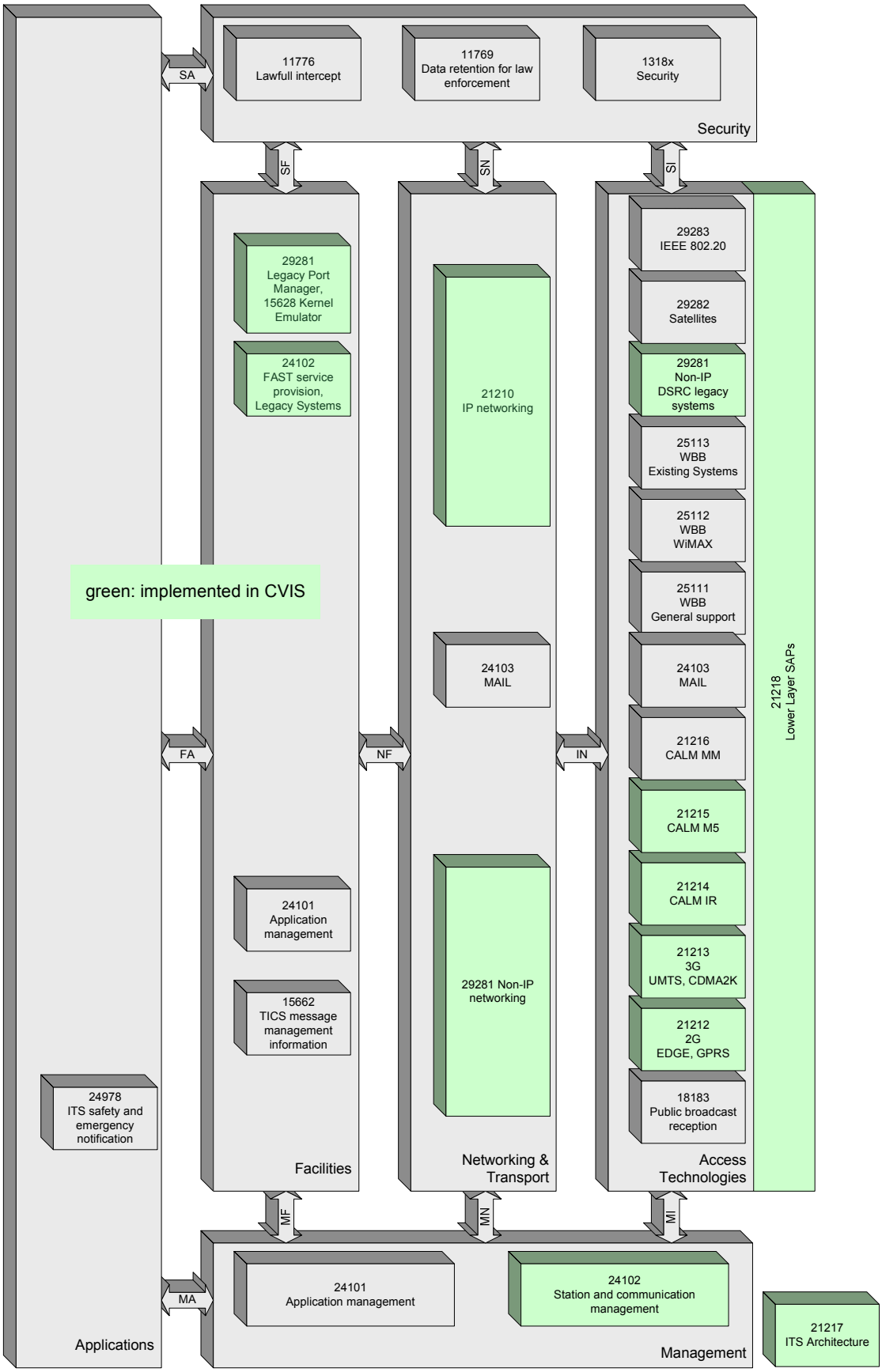

Figure 5: CALM standards

CVIS system context

The CVIS operates with existing centres, road-side and vehicle systems. Figure 6 illustrates the relation between CVIS and existing systems:

Through vehicle and road-side gateways CVIS components interface with existing systems. In addition CVIS systems use the global IPv6 network. Consequently this is a further way to communicate with other systems, e.g. existing centres.

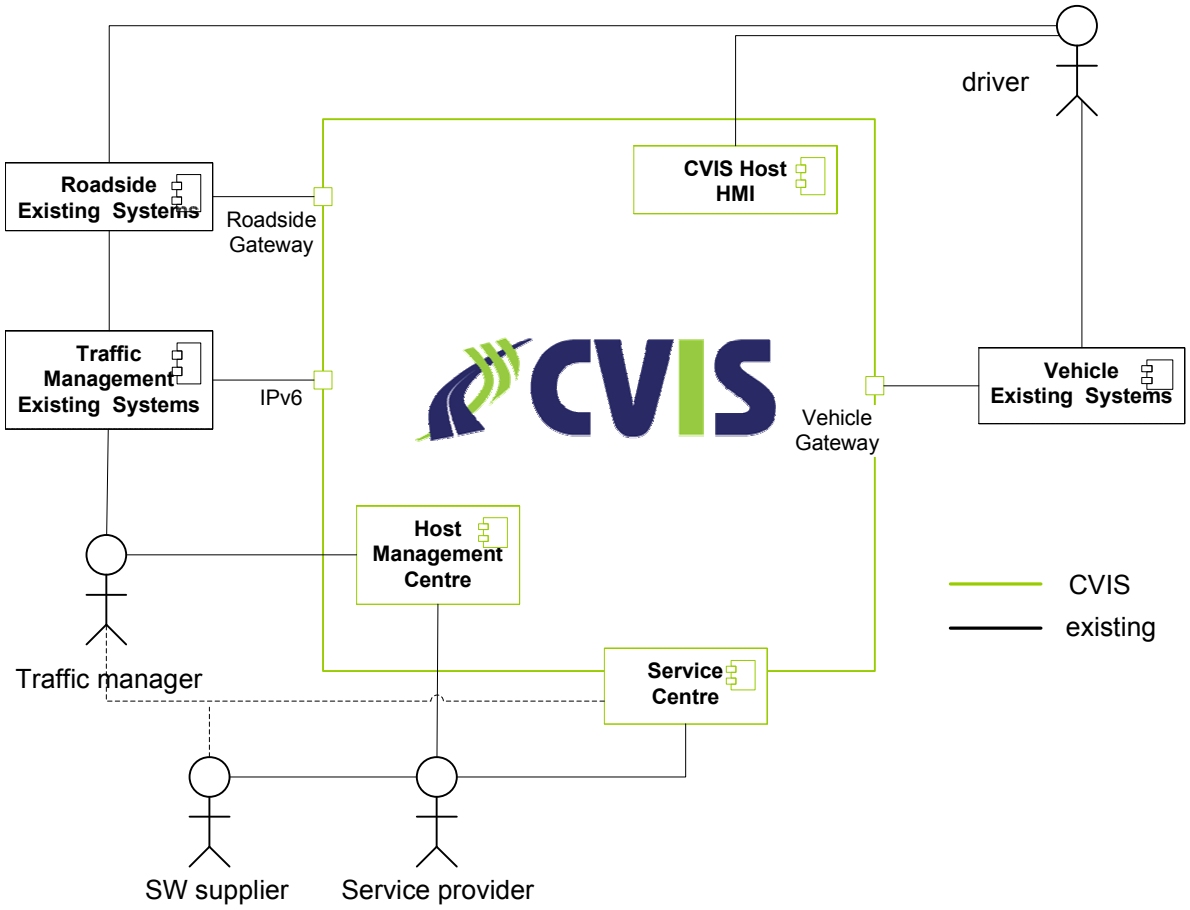

Figure 6: CVIS system context

Drivers or travellers will see CVIS through existing HMIs of vehicles, mobile devices or road-side actuator displays or applications. In addition there can be a CVIS system with own HMI towards users.

2.2 CVIS main sub-systems

A CVIS system consists of four main sub-systems as shown in Figure 7:

The central sub-system; which is the back-end infrastructure that a service provider uses to serve and operate applications and/or facilities on vehicle or road-side sub-systems. Control centres, service centres and authority databases are typical examples of central sub-system constituents.

The handheld sub-system; which provide access to the CVIS system through handheld devices such as PDAs and mobile phones. The handheld sub-system enables services such as pedestrian safety and remote management of other CVIS sub-systems. ***Note that handheld was not implemented and tested since this function is explicitly excluded from the CVIS scope.***

The vehicle sub-system; which is the vehicle *"on board"* part of CVIS system. It includes the vehicle sensors and actuators, communication infrastructures for internal communication, e.g. sensor and actuator communication, and equipment for external communication to enable car to car, car to central, car to handheld and car to road-side communication, e.g. antennas, equipment for infrared etc.

The road-side sub-system; which is the infrastructure needed to operate at a road-side unit. It can for example comprise components such as traffic lights, cameras, "Variable Message Signs" (VMS), etc.

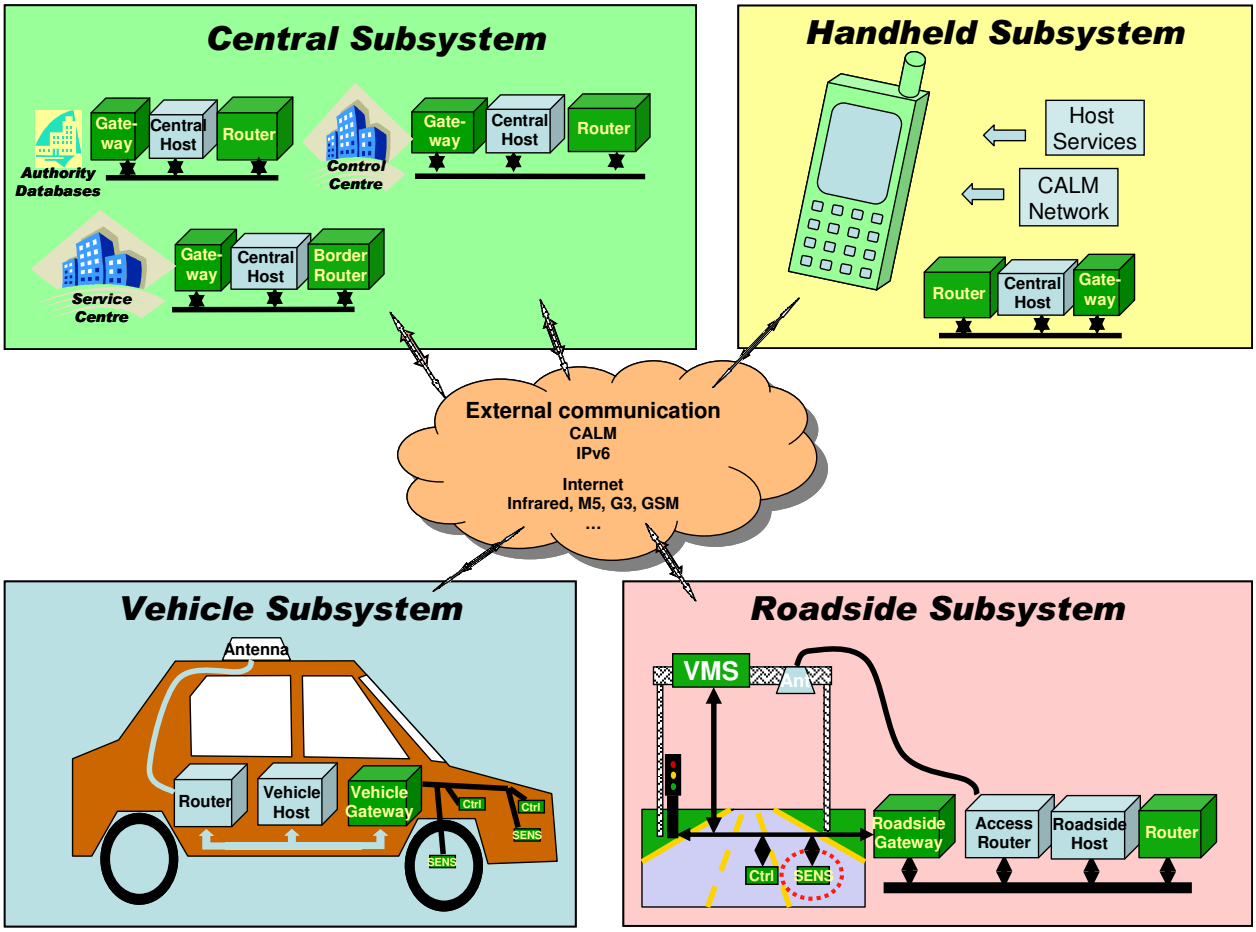

Figure 7: CVIS sub-system overview

Different networks and communication protocols are used to enable communication between different sub-systems as indicated with the cloud in Figure 7. These four sub-systems and the communication infrastructure for communication within sub-systems and between sub-systems are described further in the D.CVIS.3.2 "Reference Architecture" document.

Host, router and gateway

All sub-systems include hosts, routers and gateways as conceptual components (ISO 21217), and in the project also as physical components:

A CVIS host provides the execution environment where the CVIS applications and facilities are hosted (deployed and executed). The CVIS execution environment is based on Java and OSGi. CVIS hosts are elaborated further in the next sub-section. Applications and facilities are elaborated further in part II and part III of this document.

A CVIS router provides access to the communication infrastructure enabling connections between different CVIS hosts. There are two types of routers: the "access router" to provide wireless short-range communication and the "border router" that connects the sub-system with the Internet, e.g. cable, GPRS, UMTS.

A CVIS gateway is a protocol converter and firewall between the open and the proprietary part of a sub-system. Its purpose is to protect the technical infrastructure of the existing sub-system (vehicle, road-side or central). In a vehicle or road-side sub-system a proprietary network connects embedded controllers in the sub-system. The embedded controllers are accessible from the hosts through the gateway.

Note that all of the functionality could be implemented also in a single physical unit. A specific splitting is given by a specific implementation.

This split of sub-systems into the functional entities hosts, routers and gateways is made to separate concerns and responsibilities (the responsibilities are as described in the bullet list above). Furthermore, this split provides flexibility when configuring a sub-system since you may have several physically available hosts, routers and gateways in one sub-system. For instance a sub-system can have three hosts, one access router and two gateways. A gateway is configured to set restrictions of the access to sub-system internals, e.g. sensors and actuators. A local area network (Ethernet) connects the router, the host(s) and the gateway. The router provides communication between sub-systems.

As the CVIS host is most significant for applications developments, the following chapter is treating this component more in detail. Routers and gateways can be understood as parts of the underlying communication infrastructure, which can be used by applications residing in a host. In fact the basic facilities, e.g. connection manager and distributed directory service, provide router and gateway functions to applications on the hosts.

2.3 CVIS hosts

As elaborated in D.CVIS.3.2 "High level Architecture" a CVIS based system can be regarded as a peer to peer network of CVIS hosts, which are all connected on basis of public IPv6 as shown in Figure 8. IPv6 provides support for mobile hosts in an IP network. However when IPv6 is not available it will still be possible for connections to the network to be achieved through any available access networks e.g. 3G, Wireless LAN, IPv4 service providers.

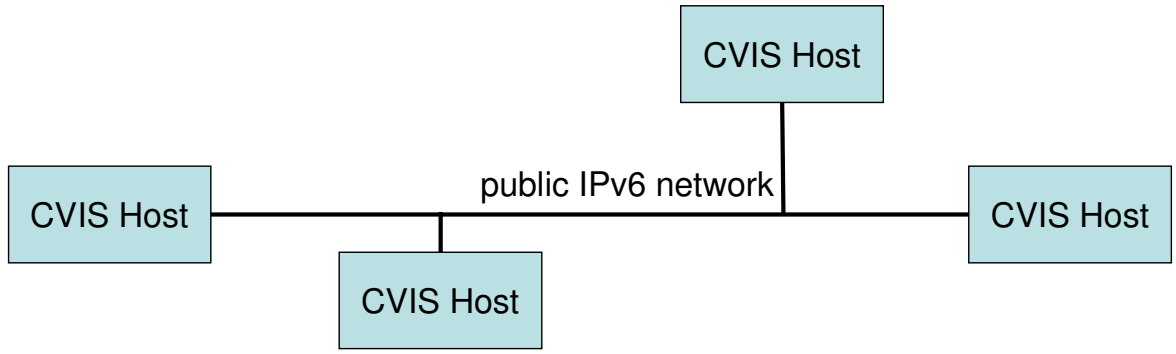

Figure 8: Network of CVIS hosts

There are different categories of hosts such as *central hosts*, e.g. control centre and service centre, *road-side hosts, vehicle hosts* and *handheld hosts* as illustrated in Figure 9.

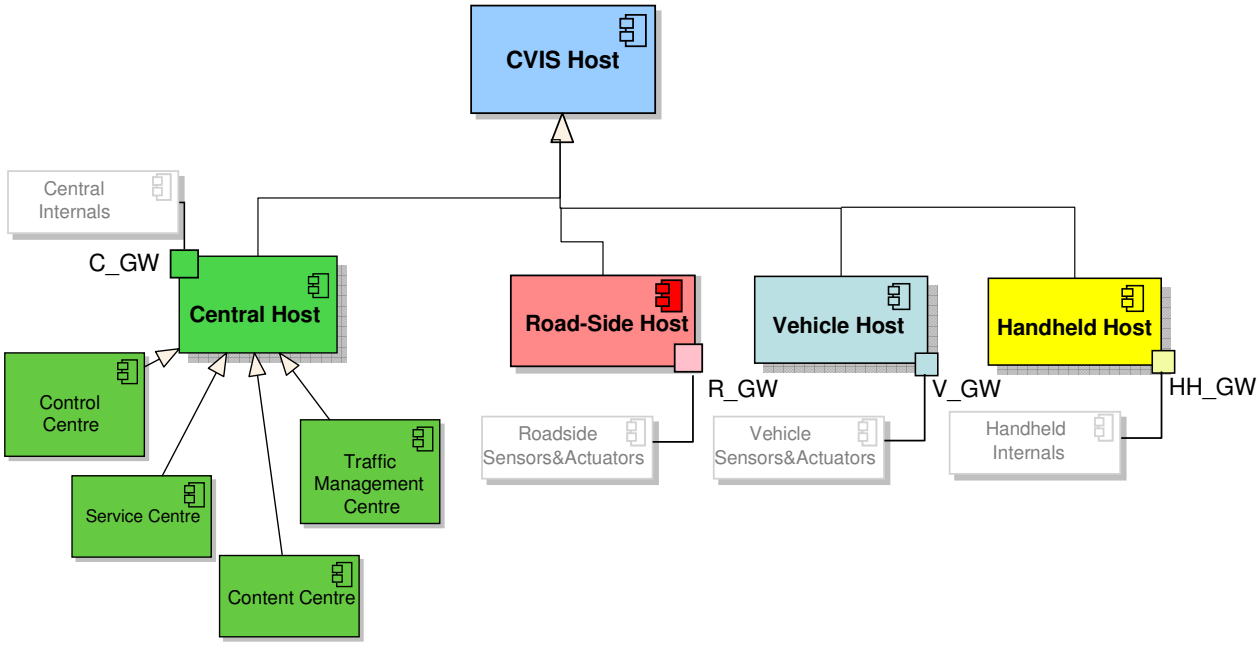

Figure 9: CVIS concept; categories of hosts

CVIS hosts can adopt various roles. An important aspect of CVIS hosts is that they can play both the role of consumer and supplier, i.e., they can supply or consume information / services (even at the same time in the same host). This is a consequence of the peer to peer architecture principle applied in CVIS. This is further elaborated in the D.CVIS.3.2 "High Level Architecture" document.

2.4 Intermediate architecture

While the described concept and architecture of cooperative systems will function in the most optimal way not until all vehicles and road-side units are equipped and are interoperable, an intermediate world needs to be considered. Full penetration of cooperative systems will not be achieved over night but sustainable intermediate model needs to be ensured.

It is obvious that if another vehicle or road-side unit is not equipped with cooperative system hosts and routers, direct peer-to-peer communication and cooperation will not be possible.

Nevertheless, some indirect cooperation could be achieved through already existing systems.

One example is "Traffic Message Channel" (TMC). If one CVIS-equipped vehicle detects e.g. an accident, slippery road conditions or traffic jams, this information could be sent to a traffic management centre which through TMC could inform other "non-cooperative" vehicles.

Another example of the intermediate world is to use "Variable Message Signs" (VMS) for informing the rest of the traffic about information provided by CVIS-equipped vehicles.

Third example is that in order to achieve better understanding of the traffic situation on a certain stretch of road, road sensors such as loops will still be needed for considerable time. The road loop data will be beneficial for cooperative vehicles as well as they will be able to receive information on road status in situations where not enough cooperative vehicles are travelling on these roads.

Last but not least, please observe that VMS and road sensors are already part of the CVIS architecture see Figure 7. Hence, even though the intermediate architecture has not been explicitly explained, it has already been considered.

2.5 Layered architecture

The overall CVIS architecture is separated into a set of layers. The layered architecture is shown in Figure 10. A main principle of a layered architecture is that a particular layer only communicates with immediate above or below layers. For example, Figure 10 specifies that the communication infrastructure is hidden from the application layer by the middleware.

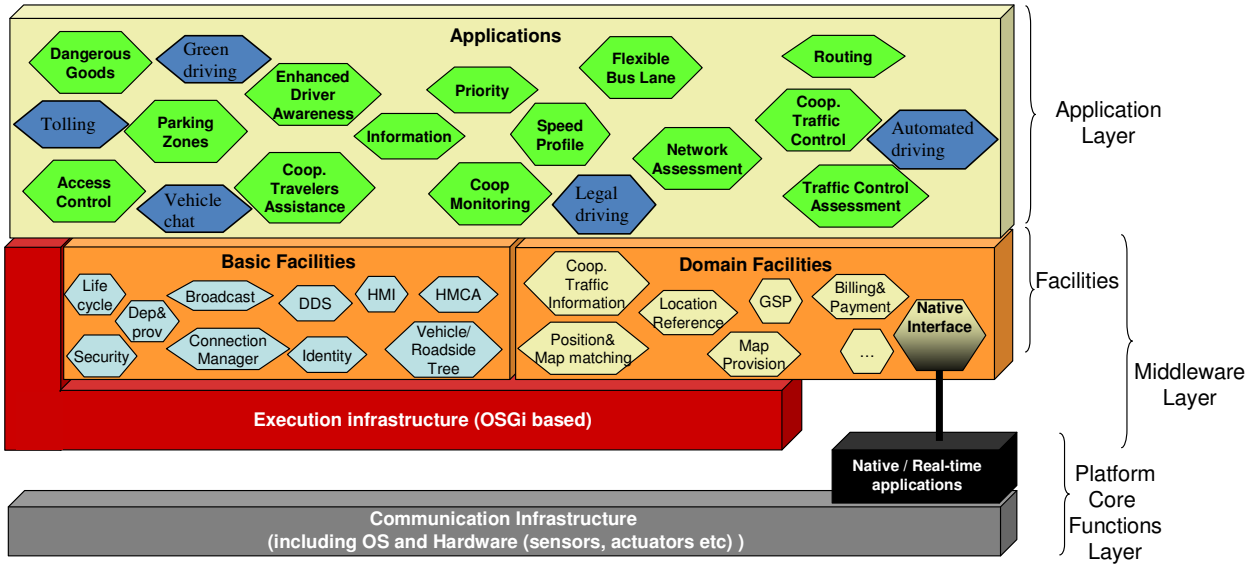

Figure 10: CVIS layered architecture

Note that in Figure 10 the term "layer" does not imply the ISO-OSI layered model for communication protocols! In addition to the CVIS layer architecture, this report also uses the ITS station reference architecture (ISO 21217) which is based on the ISO-OSI layered model as presented in Figure 99.

2.5.1 Application layer

The top layer is denoted the a*pplications layer.* It contains the set of applications. An application provides end user services. Examples of end users are drivers and traffic managers. The applications are software bundles that can be deployed and executed on the OSGi based execution infrastructure. Software bundles are presented in section 5. In the CVIS peer to peer network, applications can discover and interact with each other. During their lifecycle, applications access the basic and domain facilities in order to operate, interoperate and provide end user services. The applications are executed in the OSGi run time environment as shown in Figure 10. The set of CVIS applications (dangerous goods, enhanced driver awareness, priority etc) are presented in part III of this document. Most of CVIS applications are depicted as green hexagons.

2.5.2 Middleware layer

The *middleware layer* consists of two sub-layers. The *facilities layer* and the OSGi based *execution infrastructure layer*. The facilities layer contains a set of *facilities*. Like applications, facilities are software bundles that can be deployed and executed on the OSGi based execution infrastructure. However, the facilities provide services to support operation and interoperations of applications and other facilities. In addition, facilities can provide common domain services used by different applications and facilities. There are two types of facilities:

Basic facilities; which provide core services to support liable operation and interoperation of applications and facilities. Examples of basic facilities are: lifecycle management, "Distributed Discovery Service" (DDS) and security management.

Domain facilities; which provide common domain services such as payment and billing, positioning, map related services etc. In principle an application can evolve into a domain facility. If an application provides an end user service that also is interesting to be used by other applications and facilities, this service is a candidate for generalization to become a domain facility. The CVIS routing application is already identified as a candidate to become a domain facility.

The full set of facilities provided by CVIS is described in part II of this document.

The OSGi based *execution infrastructure layer* consists of three main parts:

1. **Java runtime environment;** which provides the execution environment for the OSGi framework.
2. **OSGi framework;** which defines an open framework that enables software installation, bundle lifecycle management, dynamic code sharing between bundles, service lookup, security, resource management, and functions necessary for remote administration.
3. **Standard OSGi services;** which add to the basic OSGi framework functionality a set of basic utility services which are considered essential for most of the OSGi bundles. These standard services are listed the D. FOAM.3.2 document.

2.5.3 Platform core functions layer

The main part of the *platform core functions layer* is the *communication infrastructure layer* which includes the communication infrastructure, operating system, routers, gateways and hardware (sensors, actuators, antennas etc). Native applications can access the communication infrastructure and hardware directly, not going via the middleware layer. Development of native applications can be required for performance reasons. Native applications should preferably provide OSGi based application programming interfaces, either at the domain facility level or at the application level. This is necessary to be integrated fully into the CVIS environment being a real CVIS *citizen,* taking part in the CVIS level interoperation, including peer to peer communication, service discovery etc.

2.6 CVIS high level component architecture

In the previous section we provided an overview of CVIS sub-systems. In this section we look into some more details specifying the content of CVIS hosts and elaborate further on the main components of the sub-systems identified in Figure 7.

A CVIS host provides an OSGi based execution environment for deployment of applications and facilities. Figure 11 depicts an example of a typical CVIS host configuration (other variations are also possible but not shown here). Two applications (a1, a2) are shown, which access different sets of facilities. The host has ports to the gateway and to a router. The gateway represents various interface mechanisms to existing legacy systems. Other facilities may also interact with the gateway to access data or execute operations, e.g. to provide information for the execution of actuators which is managed behind the gateway on legacy system side. Routers are used for communication with other CVIS hosts. There may be several hosts in a local network attached to one router.

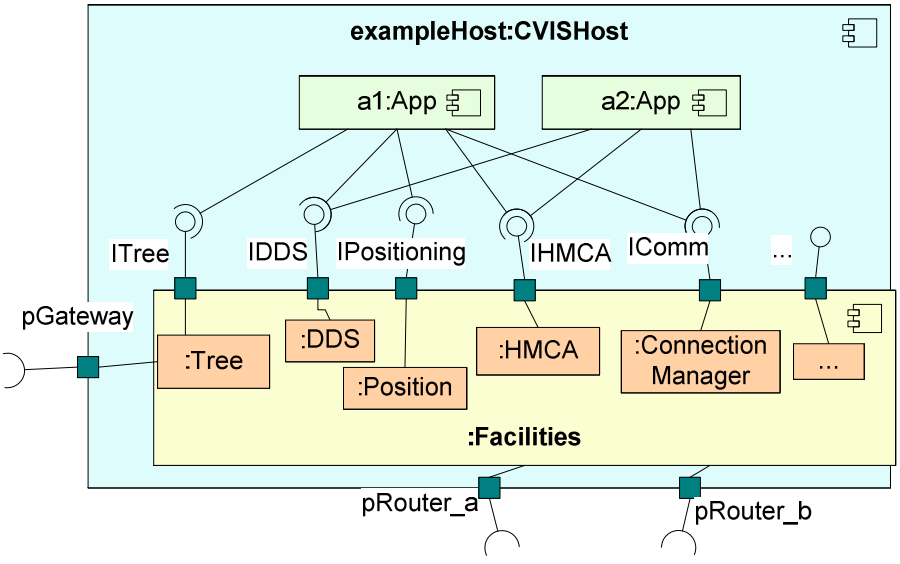

Figure 11: CVIS host

Figure 12 shows a typical configuration of the vehicle sub-system. A vehicle sub-system can contain several *vehicle hosts*. A vehicle host is a specialization of a CVIS host as shown in Figure 9. In the configuration of Figure 12 there are three vehicle hosts; v1h1, v1h2 and v1h3. The figure shows the composites of v1h1, while v1h2 and v1h3 are shown in collapsed mode. One rationale for having several hosts in a vehicle sub-system is that these may be set up with different functionality and access rights. For instance a host for executing entertainment applications and Internet browsers should have different access rights and should not be connected to the vehicle gateway. Similarly there may be several routers in a vehicle sub-system. The configuration of Figure 12 has two routers; r1 and r2. The routers provide services for communication with other CVIS hosts.

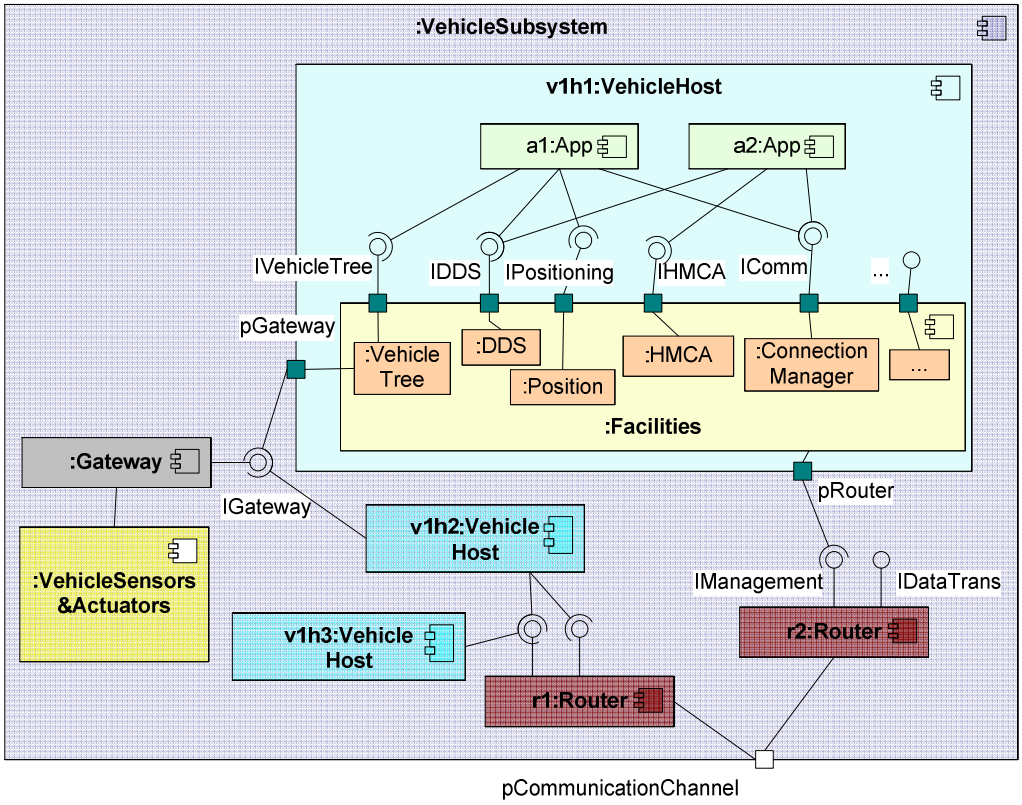

Figure 12: CVIS vehicle

The road-side sub-system has a similar architecture as a vehicle sub-system consisting of hosts routers and gateways for accessing the road-side equipment, e.g., sensors, actuators, traffic light etc. Also central sub-systems typically have the same principal architecture. This is illustrated in Figure 13.

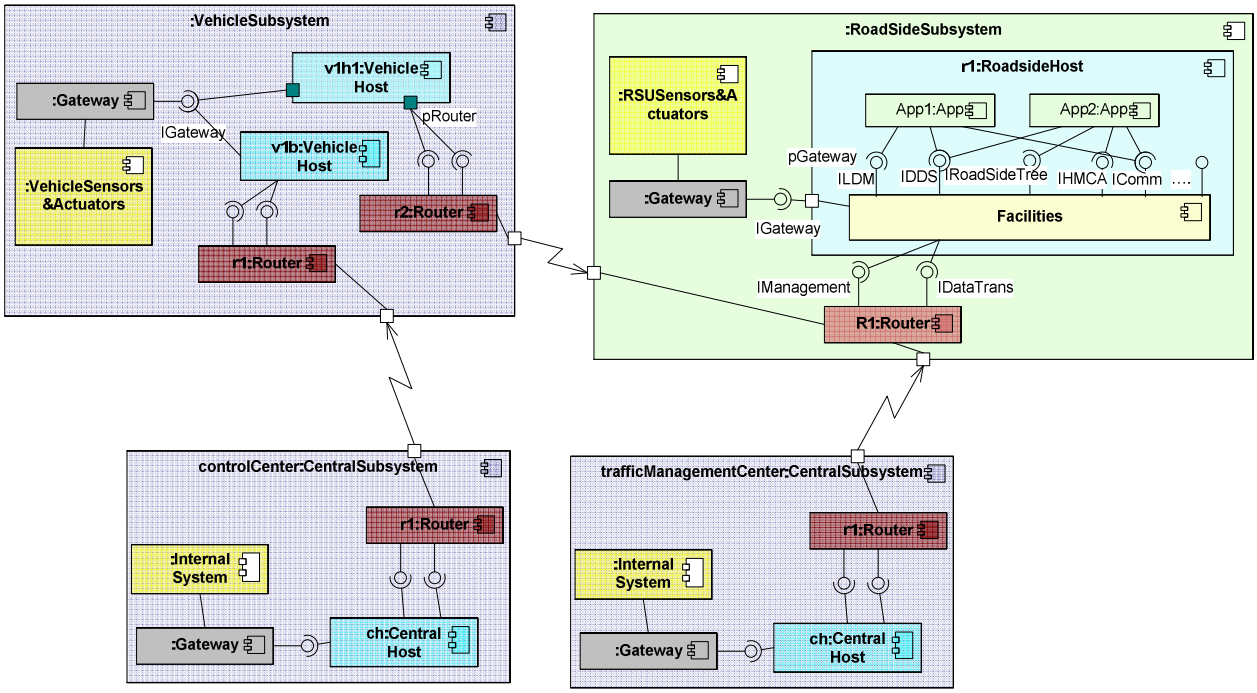

Figure 13: High level CVIS sub-system architectures

2.7 Design decisions and constraints

2.7.1 Information level interoperability

To enable interoperation between vehicles, road-side units and central units the information that are exchanged need to be understood and interpreted correctly by information consumers. Thus, unambiguous definitions of properties, types and type units of the exchanged information are important. For instance exchanging speed information of a vehicle, the type, e.g. REAL, and unit, e.g. meter per second, need to be agreed upon by the information producer and consumer.

There have been several efforts of standardizing such kind of ITS domain data, and there are several standards available such as TMC, TPEG and DATEX 2. In CVIS it is important that cooperative systems' applications use these standards as much as possible, whenever the application context allows this. However, in dedicated common functions called the CVIS "Basic Facilities" and the CVIS "Domain Facilities" as well as for broadcasts on dedicated channels, certainly a harmonised, common information modelling is necessary to achieve interoperability.

In all circumstances it is important that every information supplier provides an unambiguous specification of exchanged data. Thus, each application and facility specified in CVIS needs to include an unambiguous specification of its domain information model. The domain information model views for the different facilities and applications are included in part II and

part III of this document.

For a real world deployment of a cooperative system like CVIS develops, it is suggested to foresee an organisational and technical process that supports the sharing of information on data models inside applications. For example this could be achieved by an "Open Cooperative Systems Platform Forum". Such a forum can manage the information models, e.g. supported by a data registry mechanism.

2.7.2 Using the local dynamic map for accessing tree information

Tree information, i.e. information provided by legacy systems concerning their sensors and actors, may be accessed using the "Local Dynamic Map" (LDM). LDM enables aggregated and more sophisticated data that can be useful for a set of CVIS applications. The LDM is connected to a *DataFusion* component. The *DataFusion* is connected to the gateway accessing data available in the road-side or vehicle tree. This is depicted in Figure 14

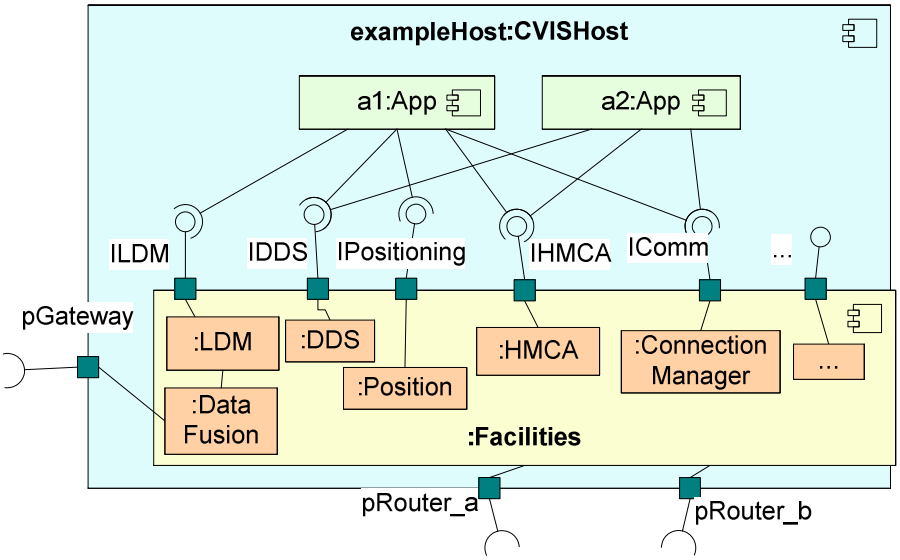

Figure 14: Access tree data using LDM

In this case the general tree component as specified in Figure 11 consists of the LDM and the *DataFusion* component. The LDM is a central component in the "Cooperative Traffic Information" facility as described in section 4.6.

2.7.3 Deployment architecture road-side

As far as traffic control and management systems are concerned, the future deployment architecture will likely look slightly different from the CVIS test site architecture. This is illustrated in Figure 15 below. The main difference between those two is that in the future all applications are foreseen to run in one environment (both CVIS and non CVIS applications), and the overall system will have the overall responsibility. In this case the sub-system foresees several application areas for different types of applications. The safety related applications (that reside in the lower green application area) and all the potential other applications (green box above) are connected to the central data base LDM which contains a geo-referenced, standardized set of data including traffic control states and aggregated detector data that is of common interest for various applications and services and is therefore available for every authorized application through standardized "Application Programming Interfaces" (APIs). Because of security issues there will also not be a direct (physical / logical) connection between the communication units or applications of the open application area and the local traffic control system. Direct control of road-side actuators is generally not allowed.

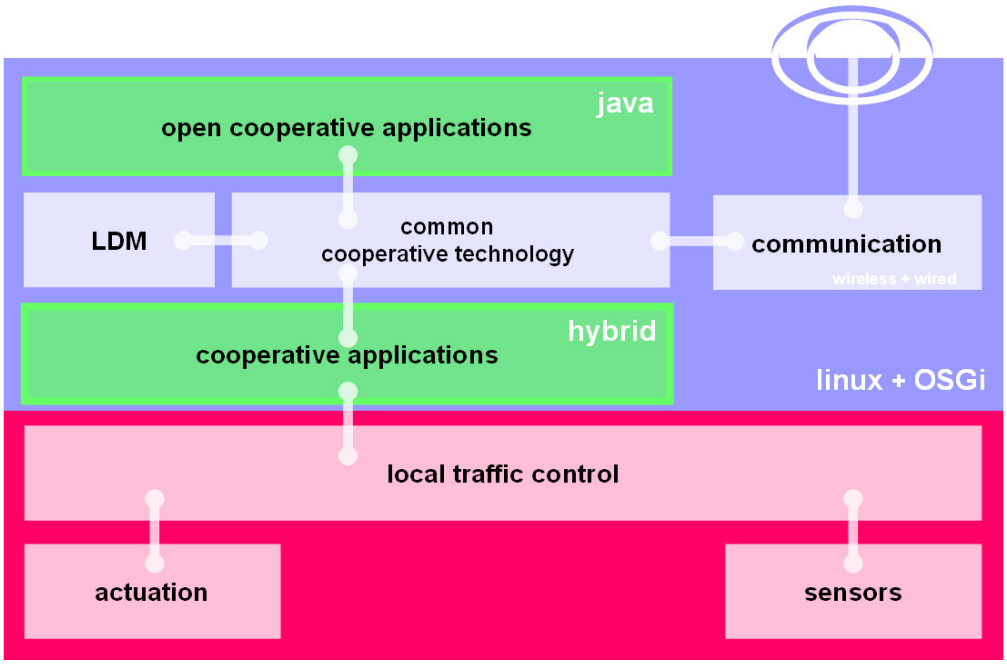

Figure 15: Co-operative road-side architecture for deployment stage

PART II CVIS facilities and infrastructure

3 Basic facilities

This section describes the set of basic facilities provided in CVIS. The basic facilities shall be available in any CVIS host. They feature common execution management and communication functionalities in the field of "Information and Communication Technology" (ICT) - whereas domain facilities in the next main chapter are common functionalities in the field of ITS.

The following basic facilities are provided:

OSGi framework and lifecycle management; which provide lifecycle management of applications and facilities.

Distributed discovery service; which facilitates looking up applications and facilities deployed in the CVIS infrastructure.

Security; which facilitate security services such as authorization and authentication and secure communication.

Identity" (as part of the security framework); which performs the identity management to allow identification with authenticated pseudonyms.

Broadcast"; which facilitate broadcasting of data and services for subscribing (to be notified of particular events).

Connection manager; which provide services for setting up communication channels between CVIS hosts.

Human machine interface; which provides a standard set of graphical elements for providing graphical user interfaces to end users.

Local device tree; which facilitates access to tree data on a CVIS sub-system, e.g. a vehicle tree, a road-side tree, etc. It also facilitates remote management and configuration of tree data.

Each of the basic facilities is presented in the next sub-sections applying the following viewpoints:

Overview"; which provides an overall introduction to the facility.

Application programming interface (API)*;* which describes the API accessible for the users of the facility (typically applications, but a facility may also be used by other facilities).

Information model; which specifies the facility from an information perspective describing information objects of the facility domain.

Interaction model; which specifies main usage scenarios associated with the facility.

High level composite architecture; which specifies the main components constituting the facility (this perspective is optional, since some facilities consists of only one main component).

This document (D.CVIS.3.4) includes specifications of interest for the users of the facilities. Further details as well as the internal design are specified in the corresponding D.SP.3.2 documents.

3.1 OSGi framework & lifecycle management

The OSGi framework provides a comprehensive set of functions to provide and deploy software bundles. This allows the addition, change or removal of applications software or facility software during the runtime of the system. CVIS will implement JAVA/OSGi in such a way, that these runtime system changes can be done from remote through the "Host Management Centre" (HMC) and the corresponding HMC on the associated hosts.

A more detailed introduction into JAVA/OSGi and the application run time environment of CVIS is given in chapter 6 of D.FOAM.3.2.

JAVA/OSGi specifications are available on www.osgi.org. e.g.

- OSGi service Platform - Core Specification, Release 4, Version 4.1, the OSGi Alliance, April 2007
- OSGi service Platform - service Compendium, Release 4, Version 4.1, the OSGi Alliance, April 2007

3.1.1 Overview

The functionality of the OSGi framework is divided in the following layers [Core]:

1. Security layer
2. Module layer
3. Lifecycle layer
4. Service layer
5. Actual services

This layering is described in more detail in section 5. Here we introduce the layers that are of special relevance for the concepts in CVIS:

The ***security layer*** is based on Java 2 security but adds a number of constraints and fills in some of the blanks that standard Java leaves open. It defines a secure packaging format as well as the runtime interaction with the Java 2 security layer.

The ***lifecycle layer*** provides a lifecycle API to bundles. This API provides a runtime model for bundles. It defines how bundles are started and stopped as well as how bundles are installed, updated and uninstalled. Additionally, it provides a comprehensive event API to allow a management bundle to control the operations of the service platform. The lifecycle layer requires the module layer but the security layer is optional.

The ***service layer*** provides a dynamic, concise and consistent programming model for Java bundle developers, simplifying the development and deployment of service bundles by de-coupling the service's specification (Java interface) from its implementations. This model allows bundle developers to bind to services only using their interface specifications. The selection of a specific implementation, optimized for a specific need or from a specific vendor, can thus be deferred to run-time.

Deployment and provisioning

Related to the lifecycle management is the deployment and provisioning features:

Deployment should be interpreted as the process of making a *service application* available at an *HMC*. This includes the packaging and transport of the application and all its components from the service centre to the HMC.

Provisioning should be interpreted as the process of enabling a *service application* for use on a *CVIS unit.* This includes packaging, transport of the application and all of its components and activation of the application on the CVIS unit.

In CVIS the host platform will be equipped with the ***management agent***. The management agent will support the lifecycle management.

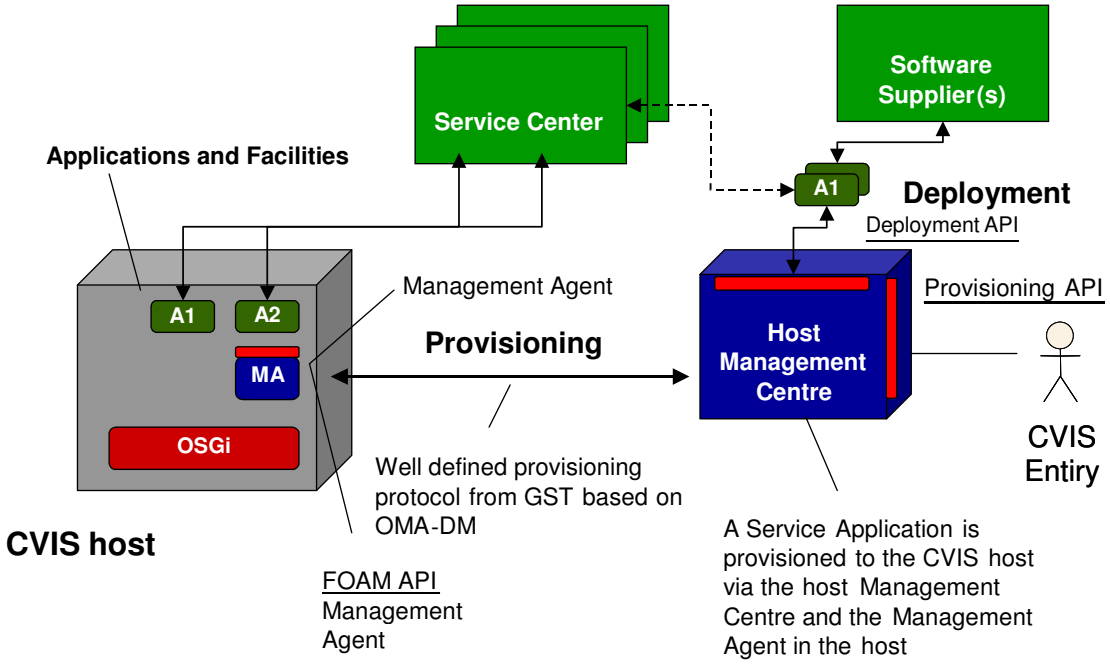

Figure 16: System overview for deployment and provisioning

Figure 16 illustrates the two different steps "deployment" and "provisioning".

- New software applications (A1, …) are issued by suppliers (or by service centres acting as software supplier) and need to be "deployed" to the HMC entities.

All CVIS hosts belong to exactly one HMC (myHMC). There may be numerous HMCs run by different organisations.

- A "Management Agent" (MA) on a host contains the necessary functionality for managing the download of new applications. This "provisioning" is based on an OMA-DM protocol which has already been demonstrated in the GST project.

After the provisioning is concluded applications (A1, A2) can run on the host. In the example figure they communicate with the service centre.

3.1.2 Application programming interface

For the actions of deployment and provisioning, CVIS defines the following Java OSGi APIs:

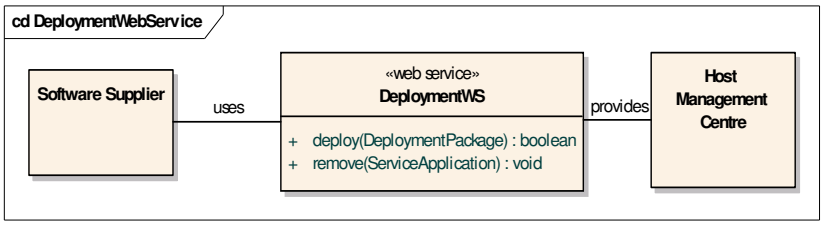

Figure 17: Deployment API

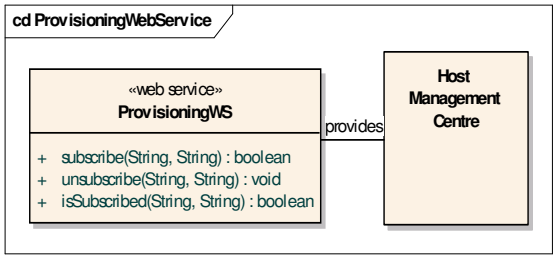

Figure 18: Provisioning API

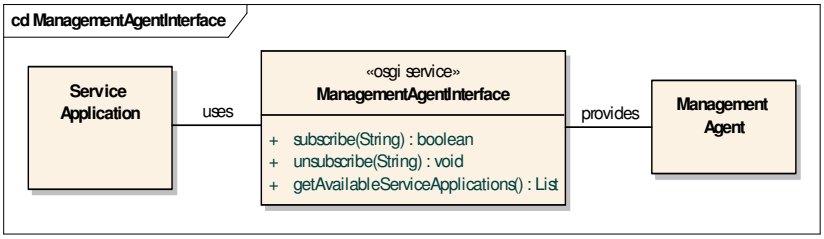

Figure 19: Provisioning API of the CVIS host

In principle here all the OSGi functionality would be right to be quoted as "the" API. As this is not the intention of this document, the following figure is just an example illustrating the lifecycle layer.

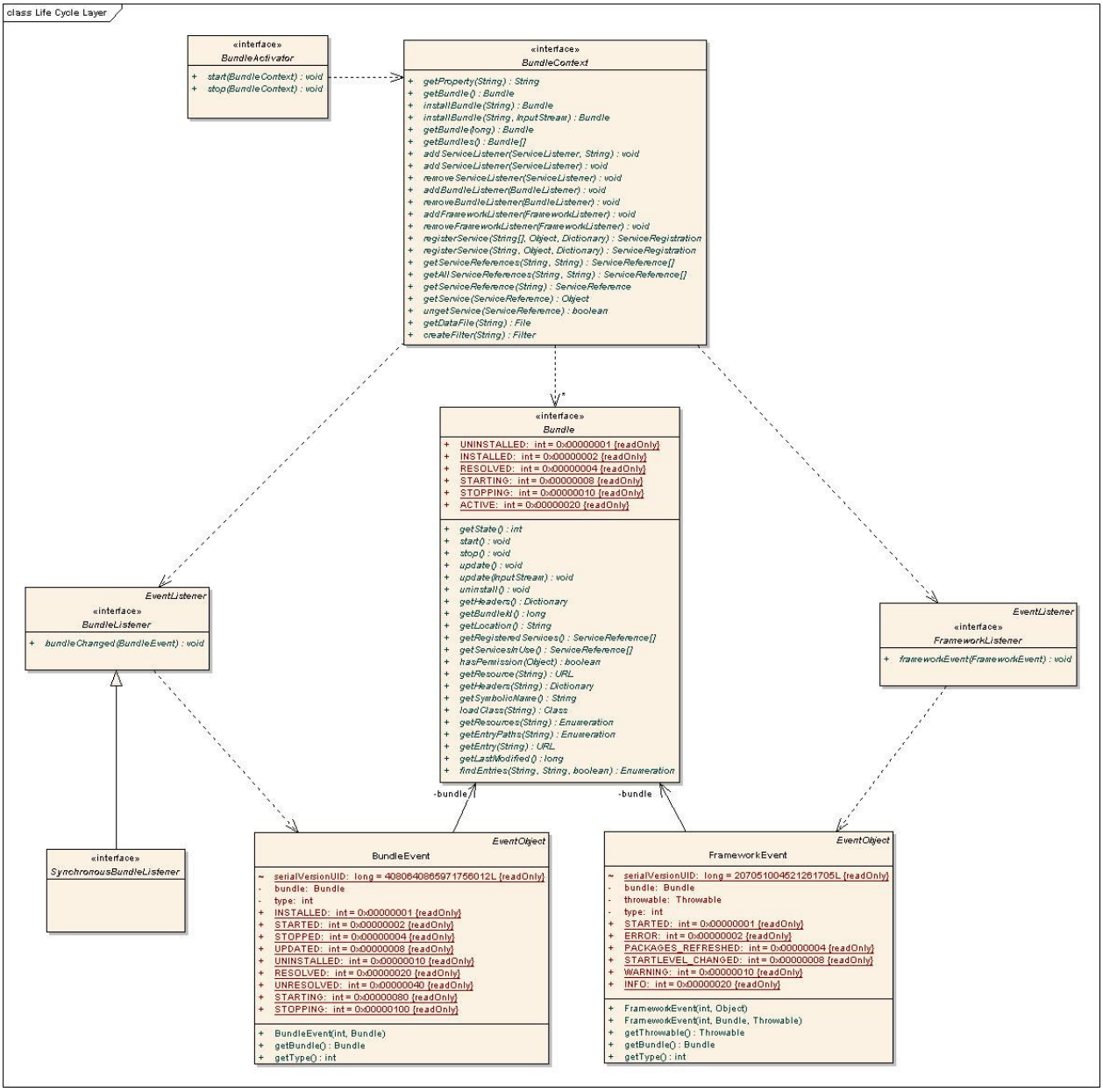

Figure 20: OSGi lifecycle API

3.1.3 Interaction model

To illustrate what the term "lifecycle" means, the state diagram of a service bundle is shown in Figure 21. As can be seen, during its lifecycle the bundle passes through 6 states:

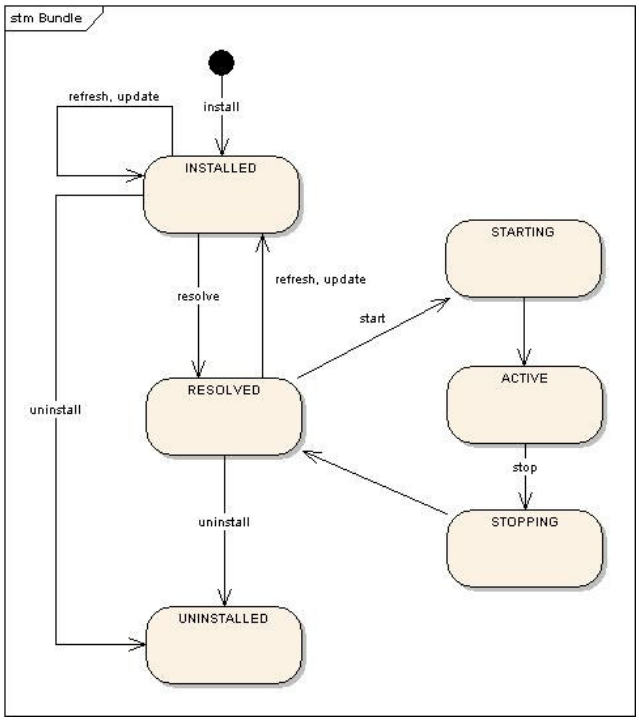

Figure 21: Lifecycle states of JAVA OSGi bundles

3.2 Distributed directory service

This sub-section is based on the "Distributed Directory Service" (DDS) section of the D.FOAM.3.2 specification document. In this document (the D.CVIS.3.4) the focus is on the external interface of the DDS and how these services are accomplished through interactions between different parts of the CVIS system. For discussions of the internal architecture of the DDS we refer to the D.FOAM.3.2 specification document.

3.2.1 Overview

DDS provides mechanisms for looking up deployed applications and facilities in a CVIS distributed peer to peer network, enabling ad hoc communication between CVIS sub-systems and applications. The DDS basically provides two discovery mechanisms:

A CVIS peer can look up another peer using some search criteria. The peers' unique identification can be used as the search criteria to look up a specific peer.

A CVIS peer can subscribe to and be notified when a particular service provided by a facility or an application is within reach. To accomplish this, the DDS provides facilities for broadcasting service announcements and to subscribe announcements of particular services. Thus, if a broadcasted service announcement reaches a particular CVIS peer that has subscribed to this particular service, the peer gets notified and they can start to interact.

Context

The CVIS system will consist of many applications and facilities (implemented as OSGi bundles) running on different CVIS hosts. A CVIS host can be a mobile unit (in-vehicle or nomadic), a road-side unit, or a centre-side system. In order to achieve their goals, applications and facilities typically need to form ad-hoc collaborations with other applications and facilities. Due to the dynamic and distributed nature of the CVIS system, the lifecycle of each application and facility will be independent from other applications and facilities. This implies that the formation of collaborations will be a dynamic process as well.

In order to establish collaboration, an application has to find the peers with which to communicate. Within a particular CVIS host, the OSGi framework offers a local service discovery mechanism. However, this mechanism is not designed for operation in a distributed, dynamic environment. For this reason, CVIS offers an additional discovery mechanism, the "Distributed Directory Service" (DDS). In essence, the DDS offers a yellow pages service across the CVIS network. This mechanism allows an application to search for applications running on other CVIS hosts based on a set of specific selection criteria. Examples of specific selection criteria are:

Applications in vehicles in a particular area;

Applications in vehicles travelling via a particular junction;

Applications in vehicles carrying (a particular class of) dangerous goods;

Applications in road-side systems in a particular area;

Applications in road-side systems along a particular road segment.

The result of the search is a set of communication handles that are returned to the searching application. Each communication handle enables the searching application to set up a communication channel to another application that satisfies the search criteria.

3.2.2 Application programming interface

DDS basic facility is provided as single Java bundle running on the local CVIS host and provides the services defined in the DDS API. The DDS API is shown in Figure 22

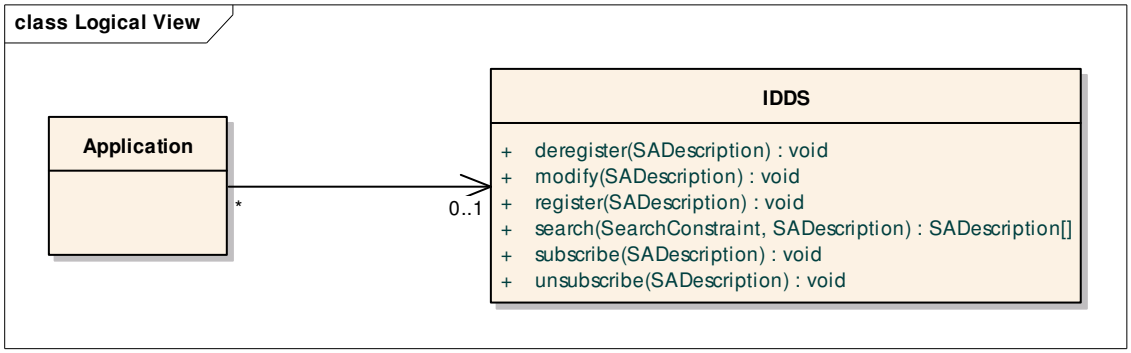

Figure 22: The DDS API

An application or a facility uses the register operation to register themselves with the local DDS facility when they are deployed. This is accomplished by submitting a description object. The specification of the description object is shown in Figure 23. It includes an identifier which is unique within the local CVIS host, a "Universal Resource Identifier" (URI)

which acts as a communications handle, some flags and an optional set of properties (encoded as key-value pairs) such as the area and direction where the vehicle is currently travelling, its cargo etc. An application can modify these registered properties with the DDS when their values have changed. It is also possible for the application to deregister itself.

The search operation is used to look up available services and applications. As part of the search an application submits a set of selection criteria by defining a number of properties. The application can also define a search constraint, which contains a timeout period (defining the maximum amount of time a query may take), the maximum number of resulting communication handles, of the maximum depth of the search. If successful, the DDS will return a set of description objects including the URIs of the matching peers. The URIs acts as communication handles. The description objects also include the last known values of the properties.

Based on the query result, the application can perform a number of actions. For instance, it can initiate a separate communication session with one or each of the matched peers, e.g. in order to perform some kind of negotiation, or it can perform a multicast to the set of peers. It could also choose not to initiate a communication session at all, but instead take an action based on the received values of the properties of the selected peers.

In the case that there is only a limited amount of discovery time available, e.g. with vehicles entering the local communication range of a road-side unit at high speed, the performance of the above DDS mechanism might not be sufficient. In that case, the publish/subscribe mechanism offered by the DDS can be used. An application can register itself with its local DDS instance by submitting a description object, where the "*isPublish*" flag is set to 'true' (its default value is 'false'). This particular form of registration implies a publish action. The DDS will initiate a continuous broadcast of the presence of the service provided by the registered application, i.e., the identifier and the URI are broadcasted within the transmission range of the corresponding CVIS unit, e.g. a road-side unit.

An application executing on a mobile host, e.g. in a vehicle, that wants to discover and initiate a communication session with providers of particular service subscribes its interest of this service. This is accomplished by calling the subscribe operation and submit a description object describing the service of interest. From this moment on, its local DDS instance will monitor all available communication channels for the presence of this service. At a certain moment, the mobile CVIS unit enters the communication range of the broadcasting CVIS unit. The local DDS will directly be made aware of the broadcasted service, and forward the description object with the corresponding URI to the matching subscribed entities. Upon reception of the description object, the recipient can directly set up a communication session.

When the application does not wish to make its services available anymore, it deregisters itself with the DDS.

Method	Type	Parameters
register(sad)	void	Sad: SADescription - in
deregister(sad)	void	Sad: SADescription - in
modify(sad)	void	Sad: SADescription - in
		Sad: SADescription - in
search(sad, sc)	SADescription[*]	Sc: SearchConstraint - in
subscribe(sad)	Void	Sad: SADescription - in
unsubscribe(sad)	Void	Sad: SADescription - in

3.2.3 Information model

The DDS information model is depicted in Figure 23.

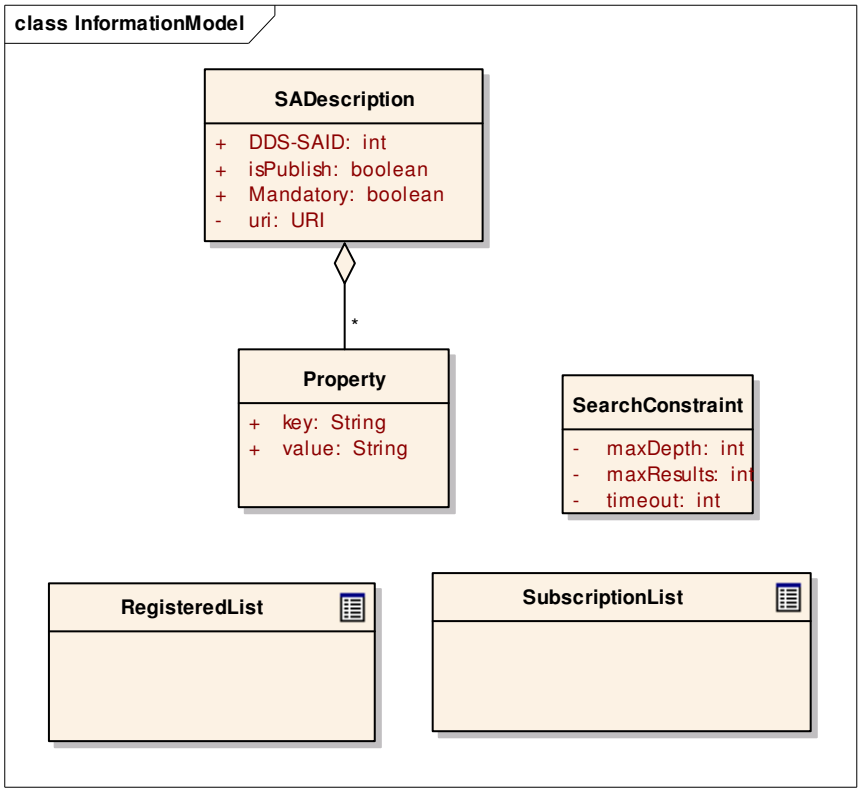

Figure 23: DDS information model

It specifies four information objects:

The description object *SADescription* contains the following attributes:

- The identifier *DDS SAID* which is defined according to element AID of the CEN DSRC standard EN 12834. This identification includes identification of the application type.
- The service announcement broadcast flag *isPublish*. Setting this flag to true implies regular broadcasts of the provided service of the registered application

or facility. The default value is false.

- The "*Mandatory*" flag. Setting this flag implies that the provided service is mandatory. For instance vehicles may be required to install a particular tolling application before entering a particular city.
- The communication handle in the form of a URI.

Properties (encoded as key-value pairs) may include information such as the area and direction where the vehicle is currently travelling, its cargo etc. The properties are used as baseline for setting search criteria when looking up particular services. Care must be taken by application designers to ensure that the property facility is used in the right way. For instance, if the application were to register the actual position as a DDS property, then a periodic update of this value (say each second) would most probably incur a huge performance cost. Therefore, properties must change its value only slowly over time.

The *SearchConstraint* defines constraints on the search. It contains the following attributes

- *maxDepth,*
- *maxResults,*
- *timeout.*

RegisteredList contains the list of registered applications and facilities

SubscriptionList contains a list specifying which local applications and facilities that have subscribed to what services.

3.2.4 Interaction model

The DDS provides two discovery mechanisms; *search* and *publish-subscribe* as presented in section 3.2.

The behavioural model related to the search mechanism is specified in Figure 24. The scenario includes a provider side (provider application, DDS, connection manager and communicationInfrastructure (CommInfr)) that resides on one particular node, e.g. a vehicle, and a consumer side (consumer application, DDS, connection manager and CommInfr) residing on another node, e.g. a road-side unit.

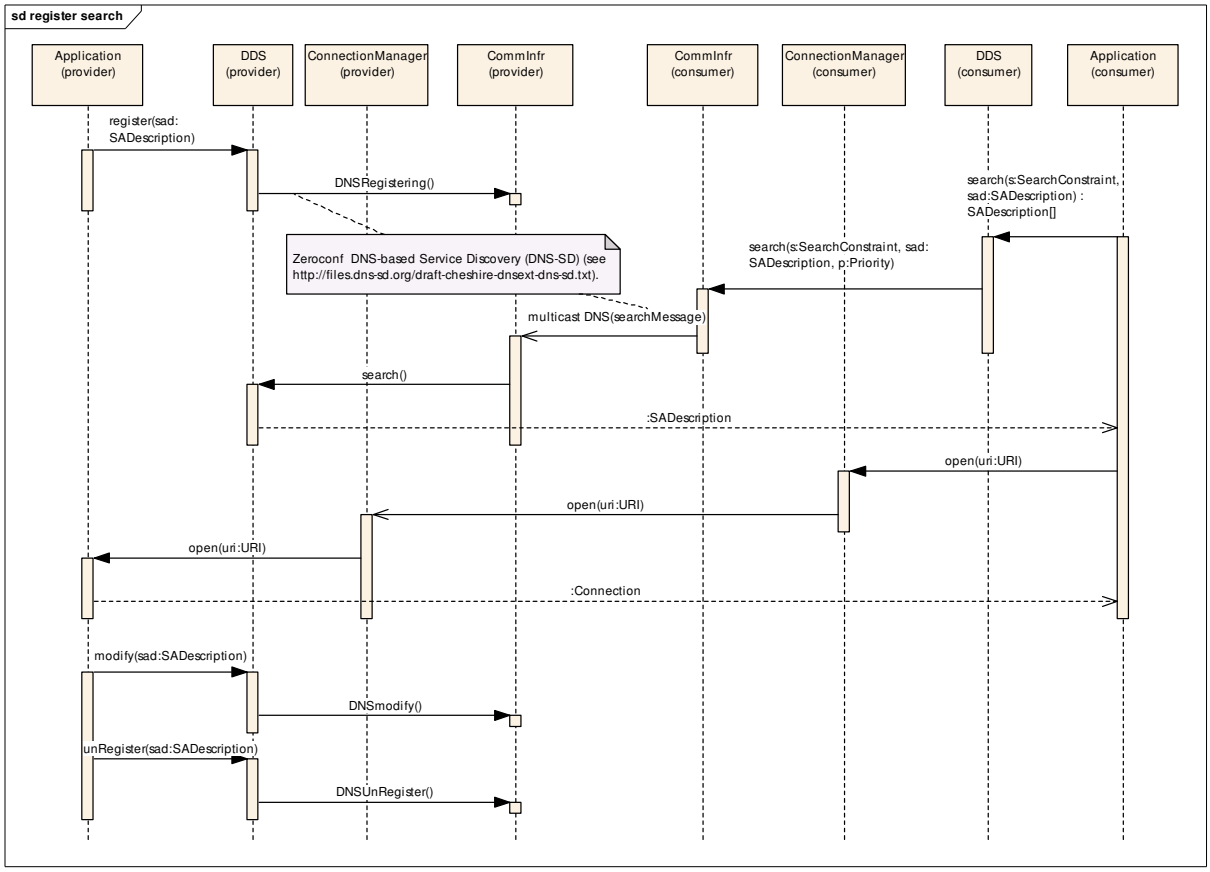

Figure 24: The search sequence model

After the consumer application has discovered the presence of provider application, it sets up a communication session through the communication handle (the URI) it has received from the provider DDS.

The usage scenario above depends on an active search to be performed by the application that wishes to initiate a communication session. Depending on the search criteria used, this search can require a considerable period of time. Since CVIS in-vehicle hosts may be travelling with speeds in excess of 40 m/s, i.e. 144 km/h, the typical dwell time within the communication range of a road-side unit with a potential communication partner may be very short (less then a few seconds).

For this reason, the DDS also offers a "publish-subscribe" mechanism. An application can subscribe to (and de-subscribe from) applications of a particular type. The DDS will then 'listen' for applications of this type, on behalf of the subscribing application. As soon as the DDS discovers, through the usage of its underlying infrastructure, e.g. CALM FAST service advertisement (ISO 24102 and ISO 29281), that there is such an application within reach, it will immediately provide a communication channel between the publisher and subscriber applications that can be used to fulfil the applications' communication needs. This is illustrated in Figure 25.

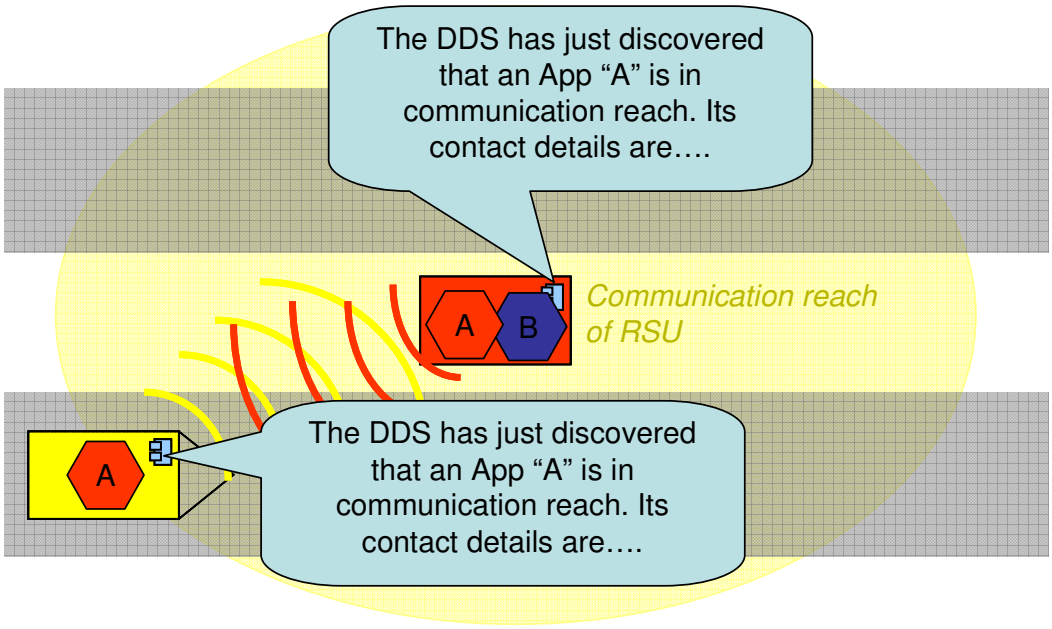

Figure 25: Illustration of publish subscribe scenario

The "publish-subscribe" sequence diagram is shown in Figure 26. Once again, the scenario includes a provider and a consumer side. After registration with the DDS, with the "*isPublish*" flag set to 'true', the DDS start continuous broadcasting of service announcements based on the information in the description object. As soon as a consumer comes within the transmission range, the consumer DDS picks up this broadcast and relays it directly to the actual application that has subscribed to this particular service. The DDS relies on services provided by the underlying communication infrastructure to provide this broadcast-and-detect facility. The connection is set up by the connection manager similarly as for the search scenario. Thus the interaction with the connection manager is not shown in Figure 26

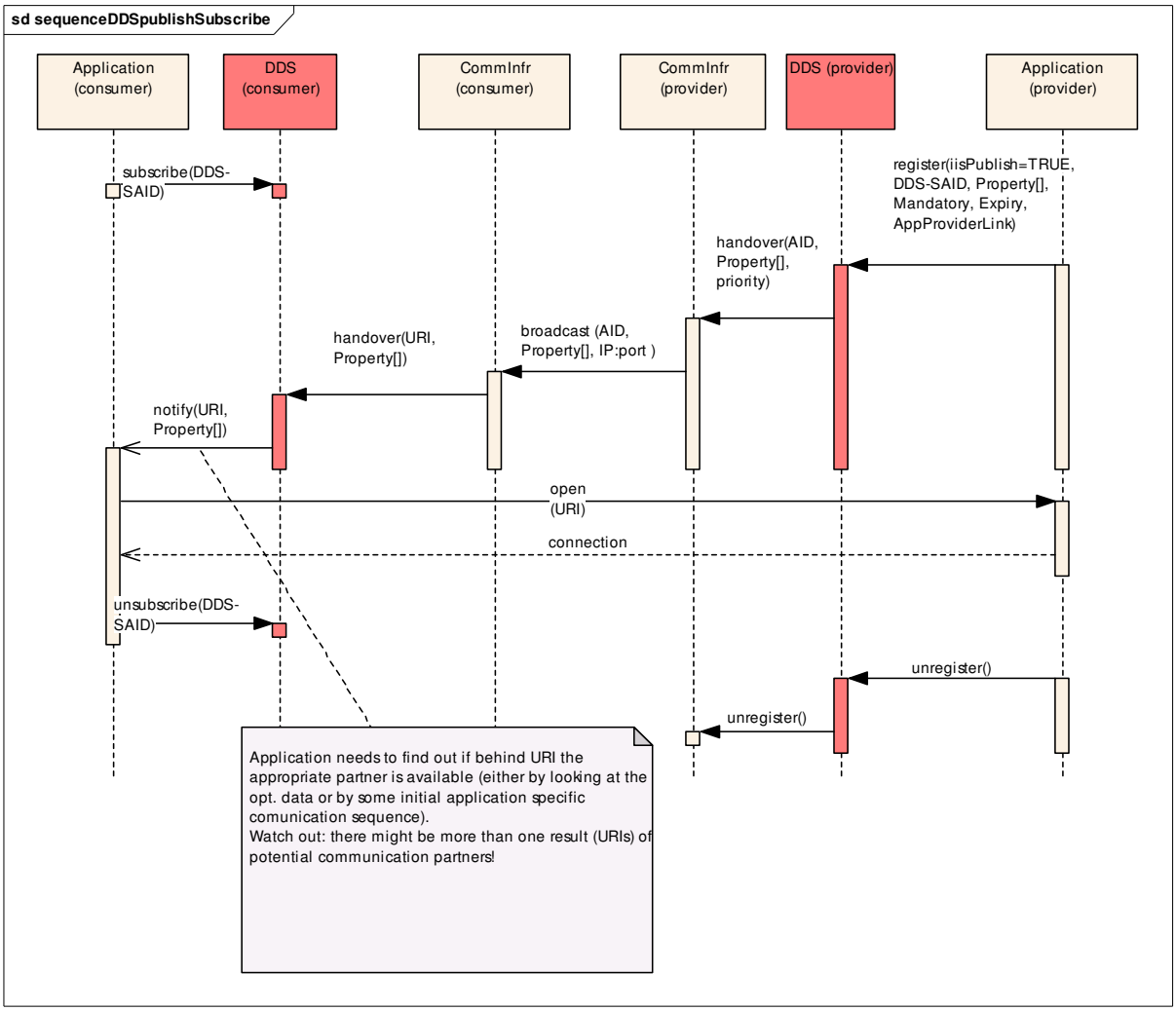

Figure 26: DDS "Publish-Subscribe" scenario

In some cases applications can be mandatory. For instance authorities of regions, e.g. a city authority, can require some applications, e.g. a particular tolling application, to be present to allow entrance into the region. When registering such an application both the "*isPublish*" flag and the "*Mandatory*" flag need to be set. The "*isPublish*" flag ensures broadcasting of service announcements and the "*Mandatory*" flag ensures that if the application is not available a download action is triggered. This scenario is a special case of the above "Publish-Subscribe" scenario. The sequence diagram is shown in Figure 27.

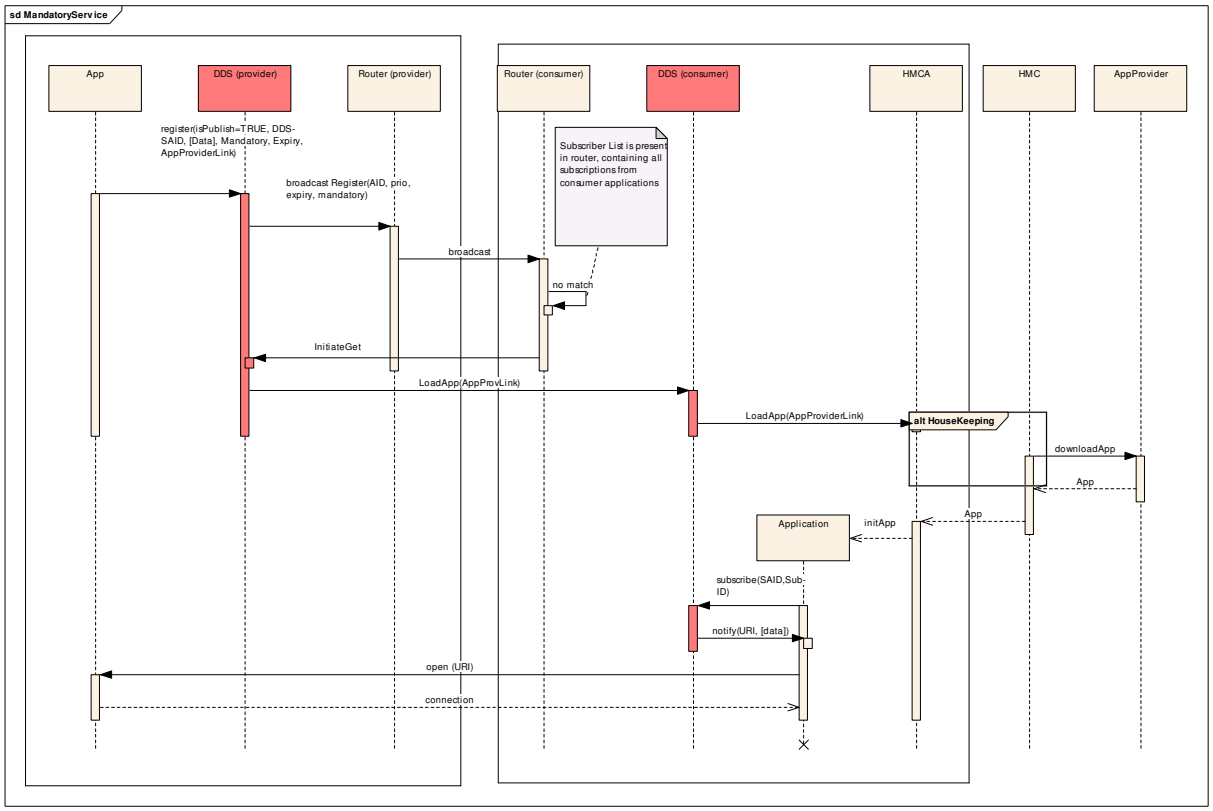

Figure 27: Mandatory service

3.3 Security framework

CVIS adopts the security framework from GST-SEC and SEVECOM ESPRIT projects. This chapter provides an overview of the security components specified. More details are described in D.FOAM.3.2 chapter 12.

It should be noted that the work in CVIS WP4 will not necessarily develop all the applications described in the architecture documents. Thus there may not be a full reference implementation, making all components available for demonstration. However the security concepts that should be used by all applications in a cooperative system are described here for completeness.

3.3.1 Overview

CVIS originally intended to build its security framework on the GST-SEC and SEVECOM ESPRIT projects. However, both projects did not deliver a working solution and CVIS FOAM had to redesign and deploy a full solution. This chapter provides an overview of the security components specified.

For application design the security framework can or shall be used by utilising the specified parts if needed for the application security. The framework offers towards applications:

Interfaces for authentication and authorisation: To be used if service applications (or an application used to log in an end user) shall approve their identity to the system and if authorisations to this identities are needed, e.g. to get access to information or to use functions.

Secure communication: To be used instead of the normal communication if a secured communication channel shall be established.

Identity manager: To be used if applications want to use specific mechanisms (e.g. pseudonyms instead of their unique, traceable identity) in order to protect privacy. (This component is under investigation in SEVECOM).

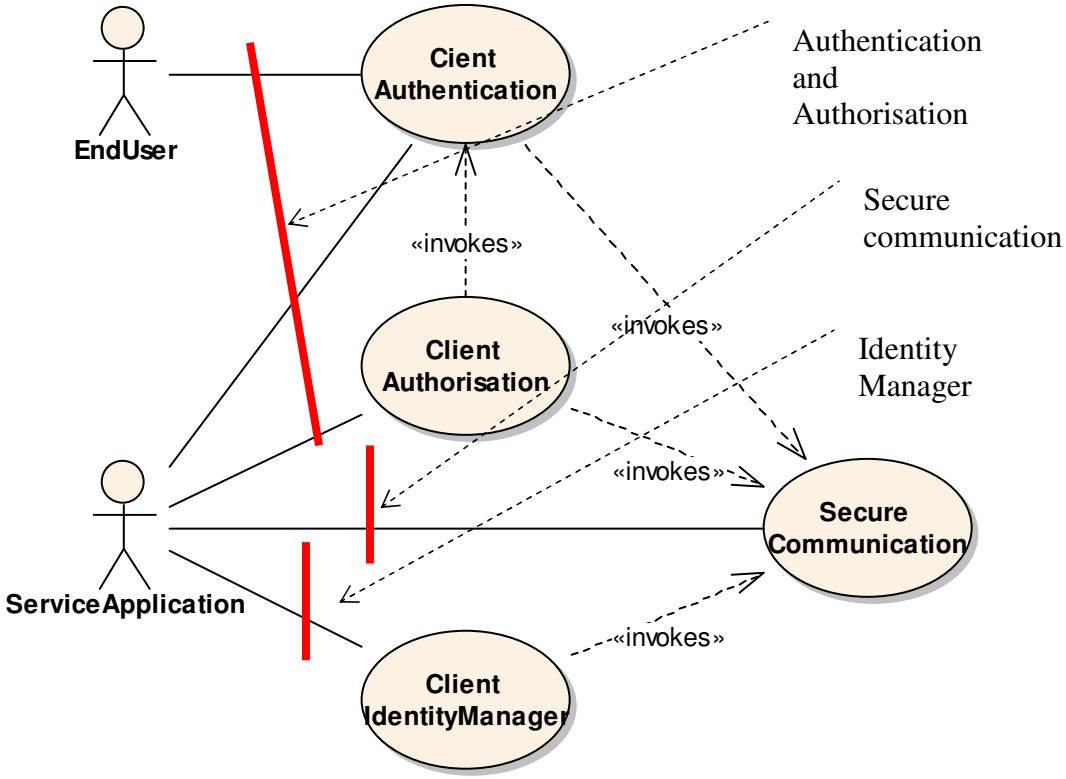

Figure 28: The security framework reference points

The security framework developed within FOAM comprises the following element:

- On the host management centre side:
  - Authentication broker
  - Authorization broker
  - User subscriptions
- On the client side (in-vehicle or roadside):
  - Authentication Broker
  - Authorization Broker
  - Secure Communication
  - Secure Module

- On the server side (application provider side and security backend):
  - Identity Manager [1]
  - Service Broker
  - Attribute and Assertion authority

The following sections will describe those elements except the identity manager (see also footnote).

3.3.2 Security framework components

1.1.1.1. Security module

Process view

The Security Module is used by the Secure Communication, and the Authentication and Authorization subsystems. The two main uses of the Security Module are:

- to provide secure persistent storage of keys, certificates and data
- to provide cryptographic functions (like signing and verification, encryption and decryption, message digests etc.).

Secure storage

Its function is to store and retrieve keys, certificates and data in a secure and persistent way.

Name	Description
Store keys	Stores different types of keys (public, private etc.) in a secure and persistent way.
Store certificates	Stores X.509 certificates in a secure and persistent way.
Store data	Stores arbitrary data in a secure and persistent way.

Cryptographic function

Name	Description
Sign and verify data	Signing and verification are necessary if data need authentication. Private keys are used for signing, public keys for verifying data. SHA-1 is used in combination with RSA.
Encrypt and decrypt data	Encryption is required in the case of data classified as confidential. Encryption and decryption are realized with the help of AES algorithms, in CBC mode, with PKCS5 padding.
Create message digest	Produces a 160 bit long hash with the SHA-1 algorithm.
Generate MAC	A message authentication code is a code produced with a secret

[1] The identity provider was not delivered by Sevecom in the course of the project. FOAM decided to use the Enterprise Sign-On Engine (ESOE, http://esoeproject.org/) as backup.

	key to protect the integrity and authenticity of a message. The MAC algorithm used in the software is HMAC-SHA-1.

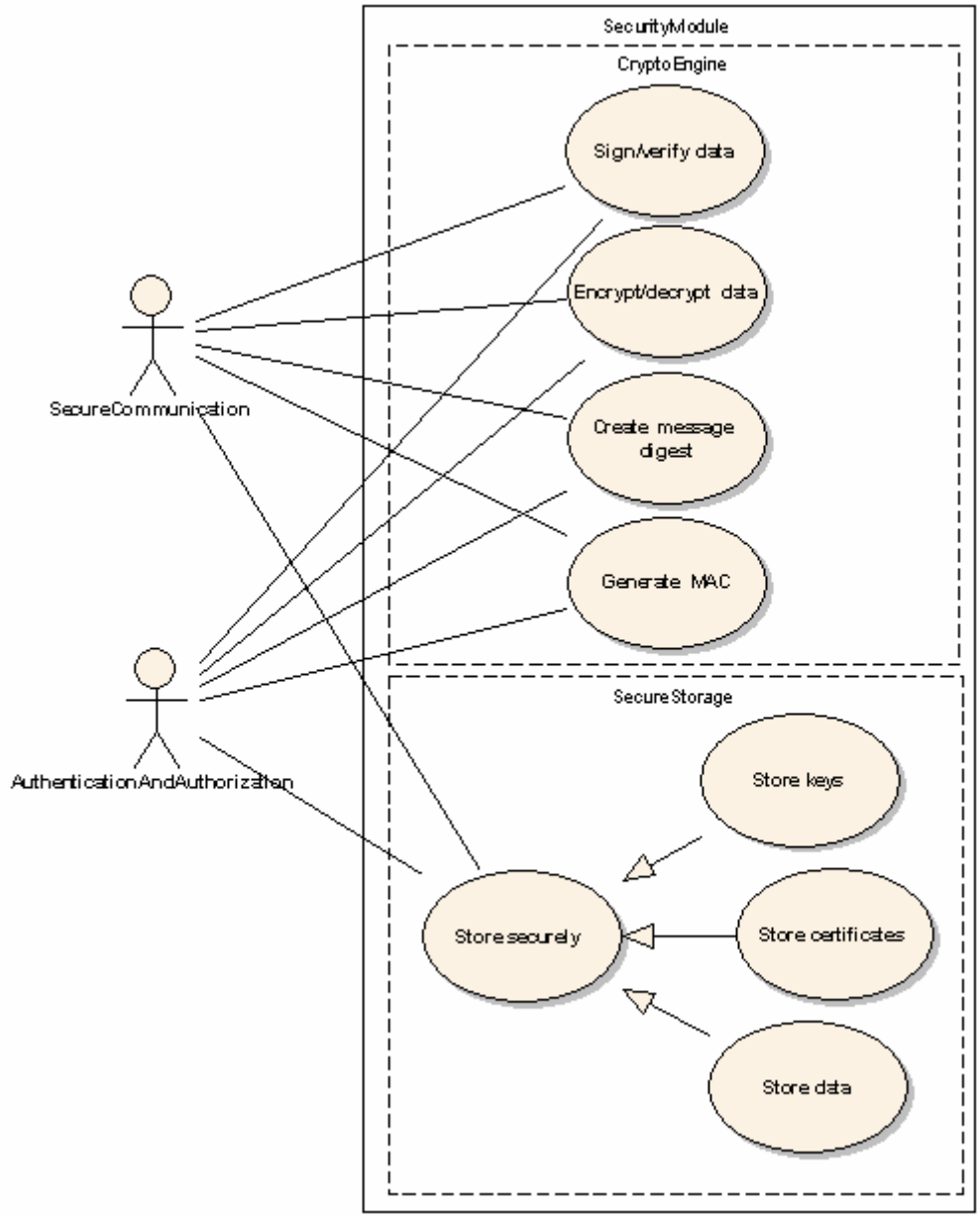

Figure 29: SecurityModule use case

Logical view

The SecurityModule subsystem can be decomposed into the following structurally significant interfaces:

- SecurityModule
- SecureStorage
- CryptoEngine
- SecurityModuleFactory
- Provider

The SecurityModule interface is at the core of the SecurityModule subsystem's architecture. It provides methods for secure storage and retrieval of data, and cryptographic functions. Therefore, the SecurityModule interface is derived from the SecureStorage and the CryptoEngine interfaces. SecureStorage is the interface that defines the methods needed for the secure storage and retrieval of keys, certificates and data. CryptoEngine's methods are for signing, verification, encryption etc.

The SecurityModuleFactory interface is the factory used for creating SecurityModule instances.

The Provider interface should be implemented by service providers who want to create their own secure storage and crypto engine implementations.

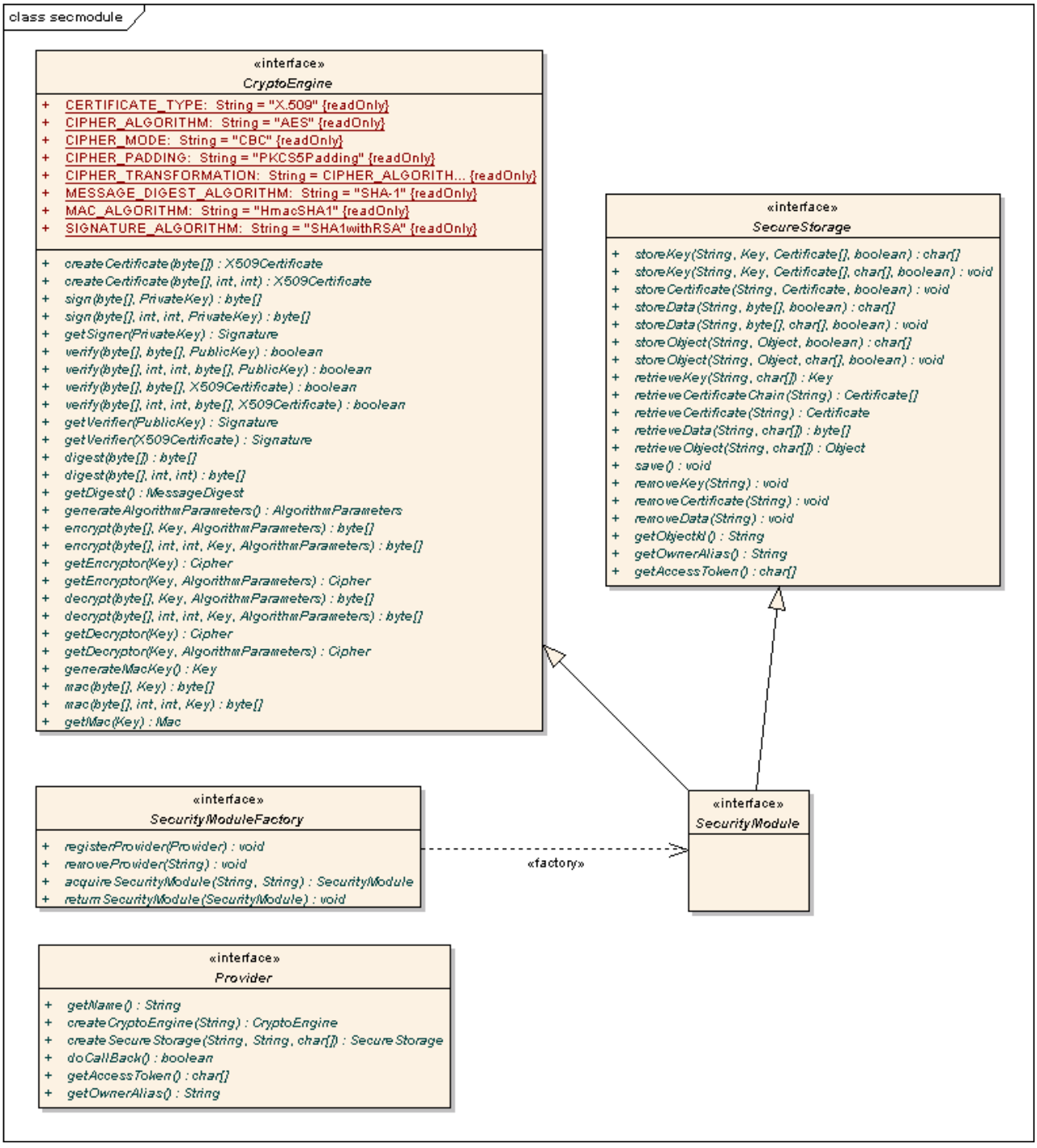

Figure 30: SecurityModule class diagram

Implementation views

The implementation classes are shown on the diagram below.

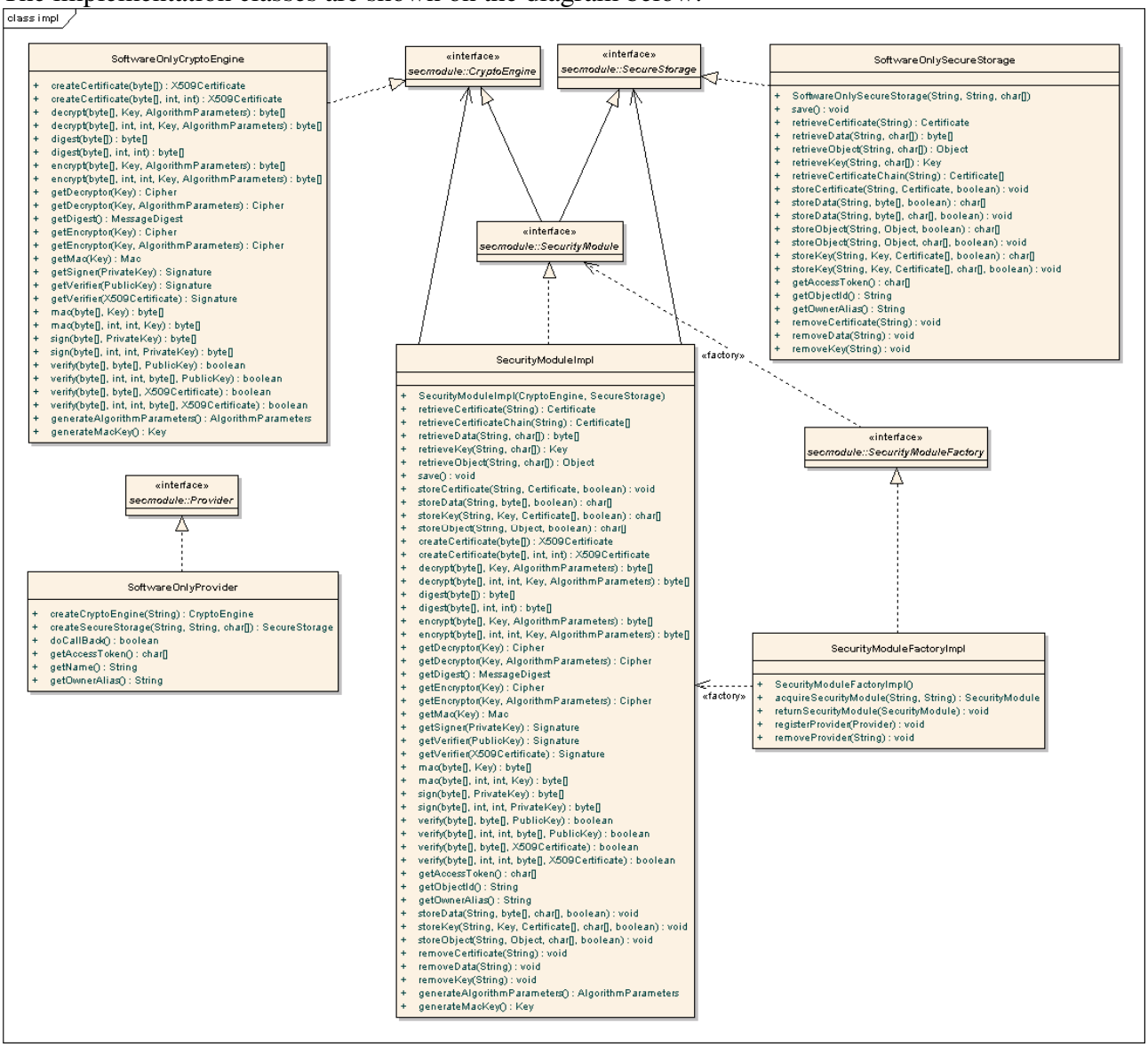

Figure 31: Implementation classes

1.1.1.2. SecureCommunication subsystem

The main usage of the SecureCommunication subsystem is the sending and receiving of messages at different security levels (insecure, confidential, authenticated and secure). Before any messages can be sent, a connection needs to be created between the client and the server, and a key agreement phase has to take place.

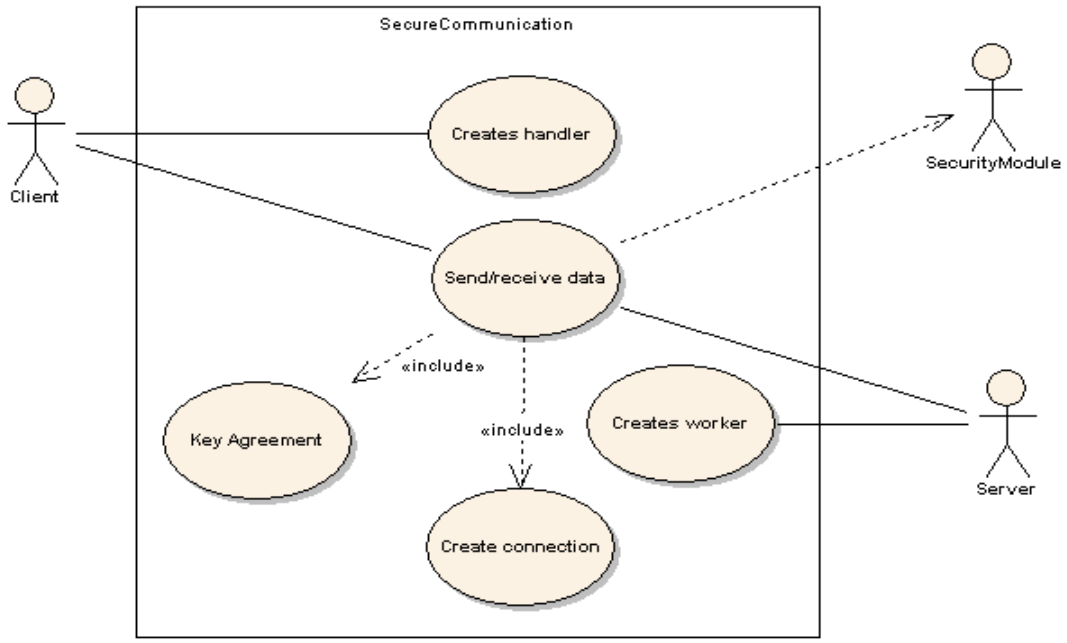

Figure 32: Communication use case diagram

Client-server setup

Name	Description
Creates handler	A handler is needed at the client side that will take care of the handling of messages. It should be created before any actual communication happens.
Creates worker	The server needs workers to deal with messages coming from or going to the clients.

Communication

Name	Description
Send/receive data	Both the client and the server can send and receive messages. Depending on the security level of the connection, the message has to be signed and/or encrypted (verified and/or decrypted) before sending (after receiving).
Key agreement	Key agreement must precede any type of communication. Its aim is to decide on common credentials that can be used later for identifying and verifying each other and the messages received.
Create connection	Establishes a connection between the client and the server, with the required security level.

Logical view

The Secure Communication subsystem comprises several interfaces, some of which are used only at the client side, some at the server side, and others at both ends. Because of the resulting complexity, first the interfaces are listed in alphabetical order, followed by the detailed description of the client and the server side.

Core interfaces and classes

Some of the below listed classes are implementation-specific, and are here only for ease of reference.

Name	Description
Handler	Defines methods to handle incoming and outgoing messages at the client side.
HandlerNotifer	A thread that will guarantee asynchronous communication at the client side.
Message	Defines the structure of messages sent through secure connections. Implementation-specific class.
MessageUtil	An implementation-specific utility class that performs the serialization of messages, reads and writes from and to channels and streams.
SCE	A secure communication engine can encode and decode messages, and initiate key agreement. It relies on the associated Security Module. Implementation-specific class.
SCEConnection	Represents a secure connection between the client and the server. A connection can have any of the following security levels: insecure, confidential, authenticated and secure.
SCEConnectionFactory	The factory that registers itself as a service and provides secure connections when necessary.
SecureKeyAgreement	An implementation-specific class that plays the key agreement.
SecureServer	The core of the server-side architecture; its role is to accept incoming connections and store them.
SecureServerFactory	A factory for creating the above mentioned server instances, parametrized according to the user's needs. Some frequently used parameters are for instance the security level of the server, the port it will listen on, and the associated WorkerFactory.
Worker	Manages messages at the server side. A Worker thread's role is to handle a message received from a client, and prepare an answer message if required.
WorkerFactory	Factory for creating Worker threads based on the user's needs. For each Worker object, a WorkerRunner object is created, which will be stored in a pool.

Name	Description
WorkerRunner	A WorkerRunner is a thread that deals with incoming and outgoing messages. More specifically, when a read or write event happens on any of the channels registered with the server, a WorkerRunner's job is to convert between Messages and byte arrays, decoding or encoding with the help of the SCE. It will delegate the task of handling and preparing the messages to an associated Worker. WorkerRunners are event-driven.
WorkerRunnerPool	An implementation-specific pool that stores worker runners.

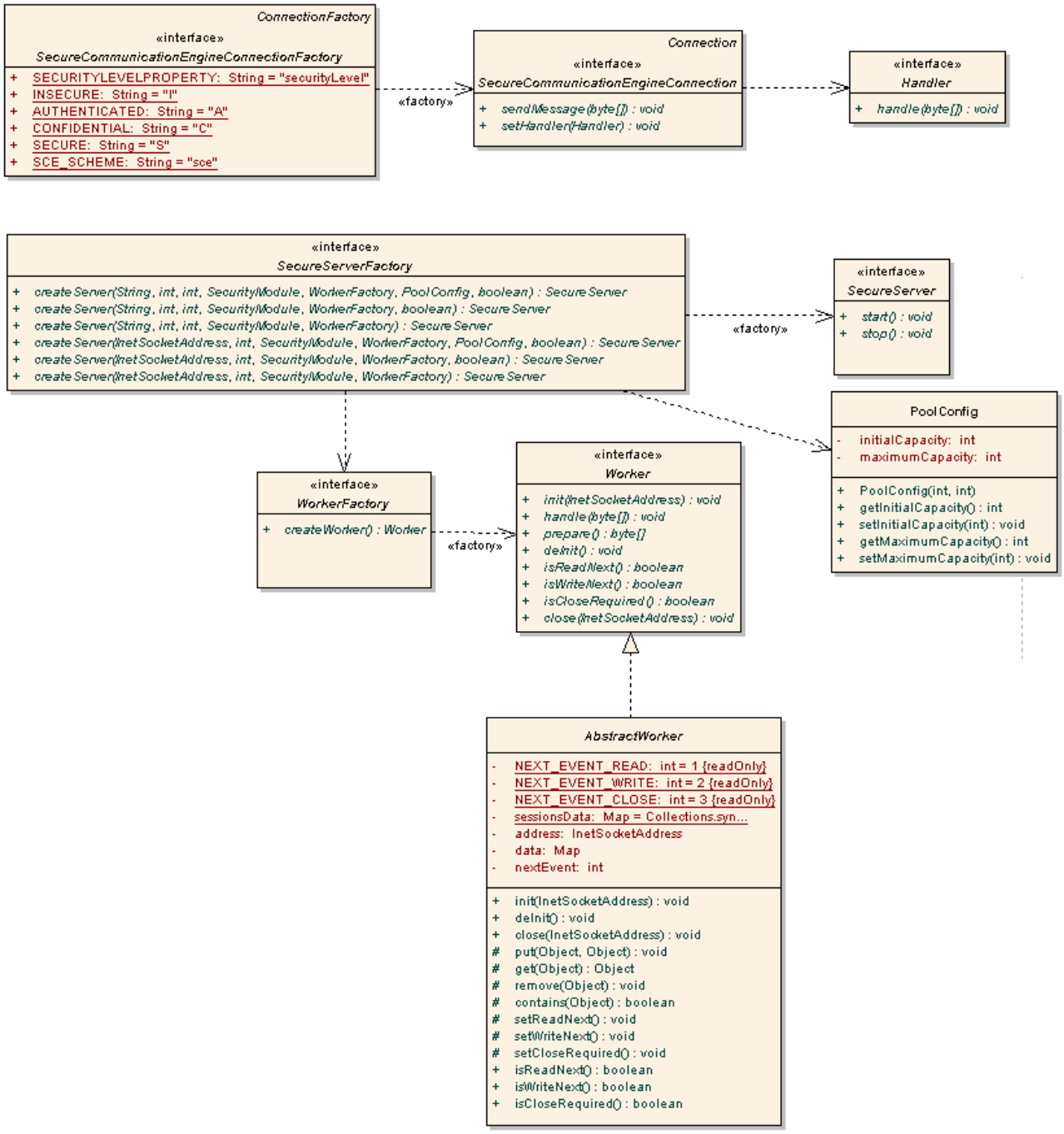

Figure 33: SecureCommunication class diagram

Client side

Connections are initiated at the client side, therefore, the SCEConnection and SCEConnectionFactory interfaces are used here. At each client, there is a SCE instance with an associated Security Module. The latter one stores confidential data that the client may require during communication (keys, certificates etc.). Messages are handled by a Handler at this end. Handlers are notified by HandlerNotifers.

Server side

A SecureServer instance is created with the help of a SecureServerFactory. A WorkerRunnerPool and a Security Module belong to each server. The WorkerRunnerPool contains the WorkerRunners that will be fetched to react to incoming messages, which partly means delegating message-handling methods to Workers. To each incoming connection an SCE is created.

Process view

Overview of the major scenarios

The following scenarios are explained in this section:

- Starting the client
- Client sending messages
- Client receiving messages
- Key agreement
- Starting the server
- Server accepting connections
- Server sending and receiving messages

Starting the client

At first, the client establishes a connection (gets a ConnectorService, asks for a proper ConnectionFactory which then creates the required type of connection). In our case, connection requests whose uri starts with "sce://" will be handled by the SecureCommunicationEngineConnectionFactory (SCEConnFactory), which will create a SecureCommunicationEngineConnection (SCEConn).

At the creation of the SCEConnection, an SCE instance is bound to the connection, and a key agreement takes place. This SCE instance will deal with encrypting, decrypting, signing etc. Messages sent and received at the client side. At the end of this phase, the input and output streams are opened that will be used for communicating with the server.

Before sending messages, a handler must be provided whose role is to handle the messages. Setting the handler triggers the creation of a dedicated HandlerNotifer thread, which will look for incoming messages.

Client sending messages

The client can send a message by calling the secure connection's sendMessage method. The required security level (I, A, C, S) is the property of the connection, but it is encoded in the message, too. First, the message data are encoded (authentication info is added, encrypted if necessary etc.) using the SecureCommunicationEngine's encode(byte[]) method, which returns an appropriately wrapped Message. As the message will be sent through a stream, which can only handle bytes and byte arrays, the message needs to be transformed into a byte array. This transformation is performed with the help of a MessageUtil class. The resulting byte array is then sent to the output stream using the stream's write(byte[]) method.

Client receiving messages

Receiving a message is an asynchronous process. A dedicated HandlerNotifer thread listens

on the input channel for incoming data. If a message arrives, the thread notifies the Handler attached to the client to handle the message. The message is retreived from the stream with MessageUtil's messageFromSource method, it is decoded using the SCE, and finally the Handler's handle method takes care of dealing with the message.

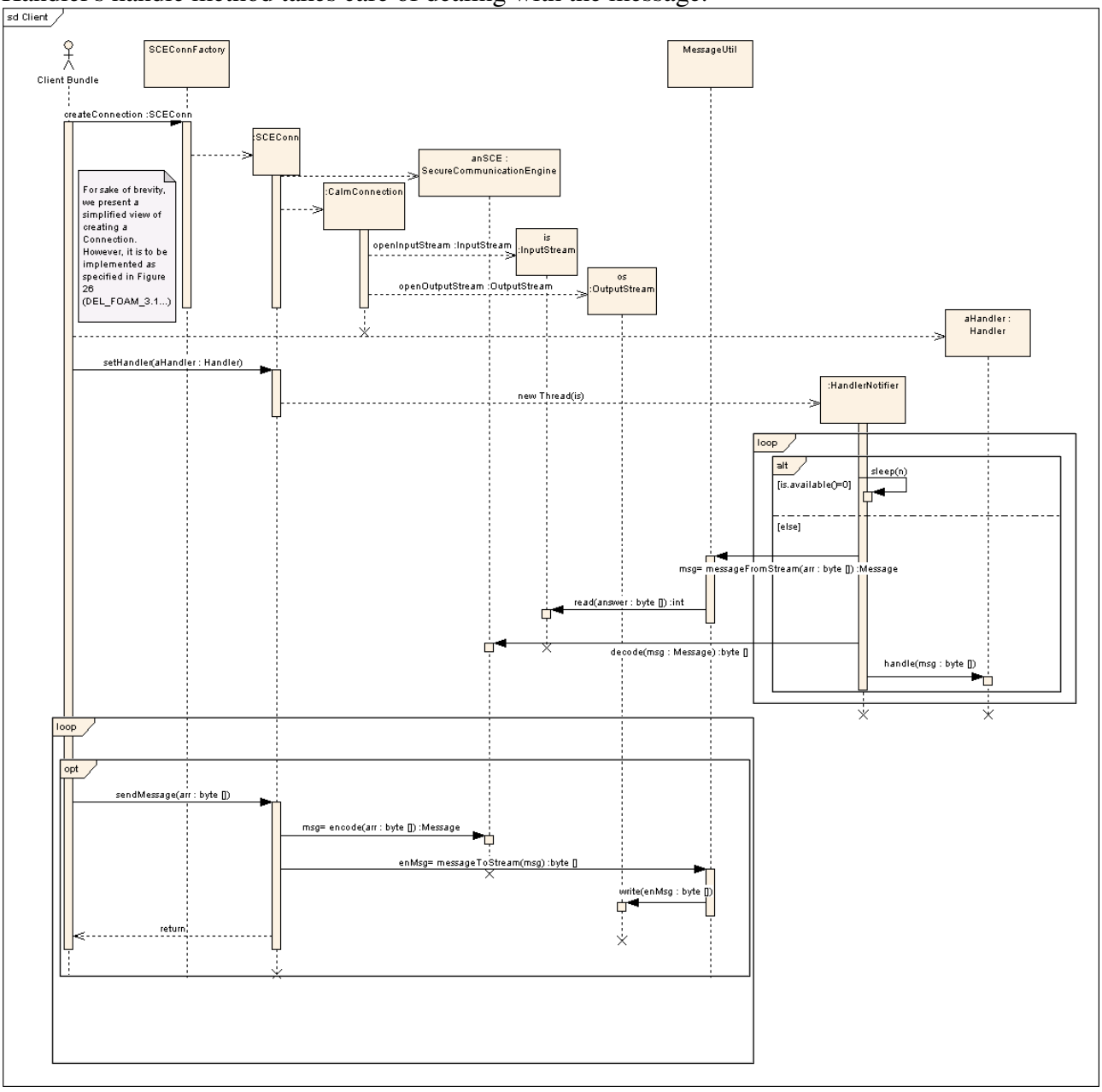

Figure 34: Creating connecions, sending and receiving messages at the client side

Key agreement

Before the client can send messages to the server, a key agreement process has to take place, the aim of which is to agree on the credentials that will later be used to secure the communication. Key agreement has to take into account the required security level. A connection can only be created if the security level of the client is the same as that of the server.

It is always the client that starts the key agreement. First, it sends a session ID to the server

which either accepts it or creates a new one. This depends on whether the server has the relevant information about the client stored from previous communications. If the security level of the end-points is not the same, communication fails and the connection is refused.

Further action depends on the required security level.

- *Insecure connection (I).* Insecure connections do not require that messages be encrypted or signed, therefore, key agreement ends, and the connection is established successfully.
- *Confidential connection (C).* Confidentiality means that messages are encrypted. This requires that both partys have a common key to encrypt and decrypt messages. The client sends random data to the server that generates its own random data based on what it has received, and sends it back to the client, together with its certificate chain. On the basis of the data received from the server, the client generates an Initialization Vector and an AES key, and sends them to the server. The client then creates a hash from all the data it has sent with the common key. The server receives the hashed data and decrypts it with their common key. If the decrypted data are not the same as those it received earlier, the server aborts the connection. Otherwise, the server signs data it has sent earlier with its private key. Finally the client verifies the server using the public key it has received from the server in the certificate. If verification fails, the connection is refused. Otherwise, the connection is established successfully.
- *Authenticated connection (A).* Authenticity requires that the client and the server have each other's certificates that contain the public keys needed to identify the other party. At the beginning, the client and the server send their own certificate chains to each other. They recover the other party's public key from the certificates. The client and the server then sign their own certificate chains with their own private keys, and exchange the signed data. Finally, they verify each other. If verification fails, the connection is refused. Otherwise, the connection is established successfully.
- *Secure connection (S).* This type of connection is the combination of the confidential and authenticated type. As such it requires that the client and the server agree on a common key , exchange certificates and verify each other as described above.

Starting the server

The server as such is an OSGi bundle. It seemed reasonable therefore to implement the server's setup at the Activator's start method, while the shutdown in the stop method.

At server startup, the user creates a SecureServer instance with the help of the SecureServerFactory, where it can specify the port that the server will listen on, the security level of the server, the associated Security Module, the WorkerFactory that will create the Worker threads, and the pool configuration properties. The latter one is optional: if no PoolConfig object is given, a default setting is used. It can also be specified whether the server's first task should be receiving messages from clients or sending out messages.

Before the actual SecureServer is created, a new worker pool is filled with the number of worker runner threads specified in PoolConfig. This happens by calling repeatedly the WorkerFactory's createWorker methods, and instantiating WorkerRunners with the returned Worker threads. At the beginning, all workers are available.

The SecureServer is instantiated with the specified port and the previously created worker pool. Calling its start method will start the server.

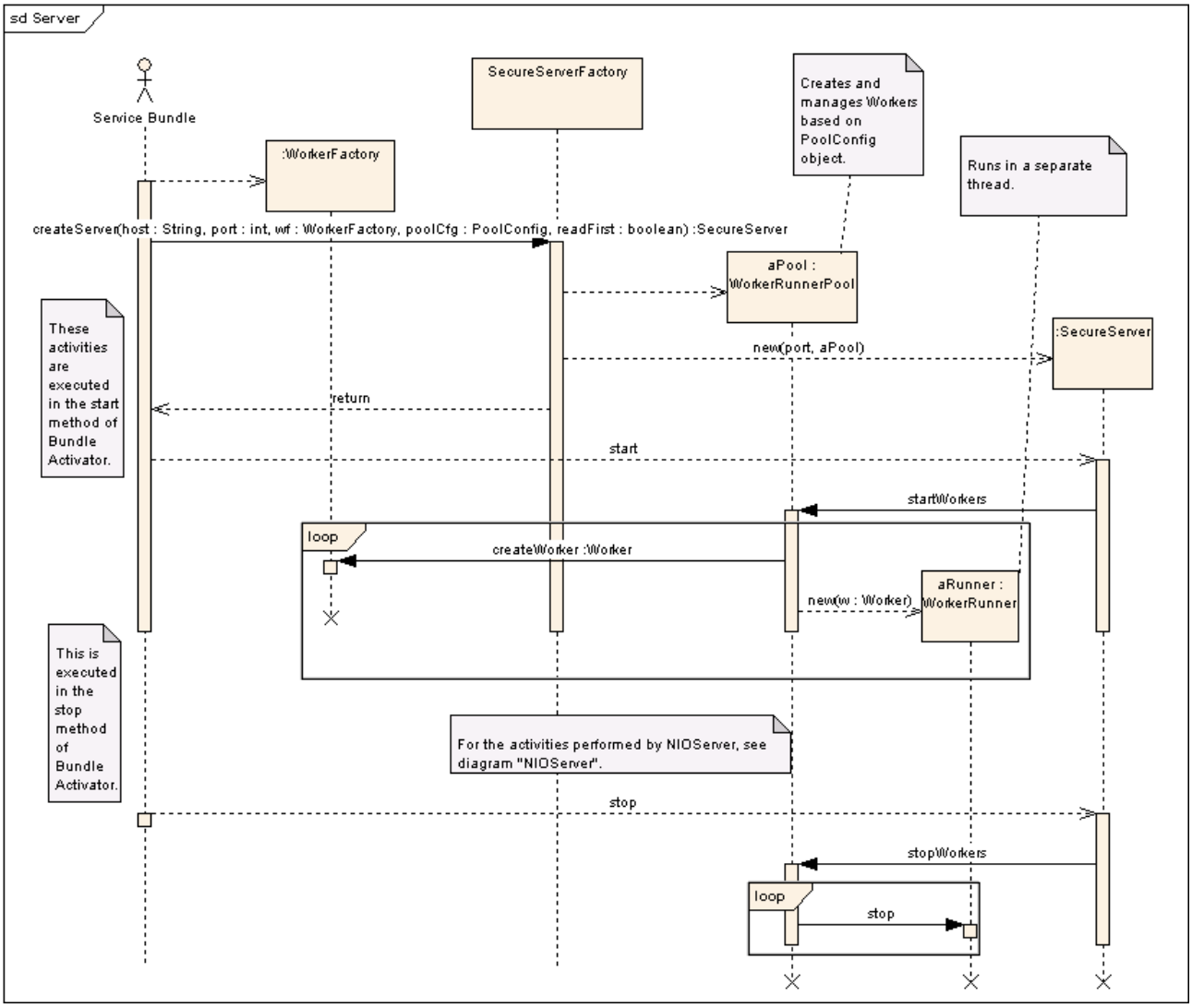

Figure 35: Server start-up

Server accepting connections

The SecureServer's job is to listen for incoming connections on the port by calling the underlying ServerSocket's accept method. Channels belonging to the clients register themselves on the server's Selector. The SelectionKey objects that hold this registration will be used to identify clients.

Server sending and receiving messages

The SecureServer will check repeatedly whether there are read or write events on any of the registered channels (this is accomplished by iterating through the selected SelectionKeys).

If there is a read event, i.e. a message arrives on a channel, the server fetches an available WorkerRunner from the pool to do the task. First, the WorkerRunner opens the InputStream associated with the channel belonging to the specified SelectionKey. The WorkerRunner can then read the message in the form of a byte array from the stream. The Message object is retrieved and decoded as has been described in the case of clients (i.e. relying on MessageUtil and SCE). The decoded message is transferred to the WorkerRunner's Worker, which will handle the Message.

Two cases are possible at this point:

- If the message requires an answer from the server, the worker does not retire to its pool immediately. Instead, it sets its writeNext parameter indicating that it needs to prepare an answer and send it to the client.
- Otherwise, the WorkerRunner is ready, it retires to the pool to become available for use again if needed.

If a message needs to be sent to a client, an available WorkerRunner is fetched from the pool. The associated Worker thread prepares the answer, which is then encoded, serialized and written on the appropriate output stream. At this point the WorkerRunner may indicate that it will not have to deal with this connection any more, so it can be closed. After the job is accomplished, the WorkerRunner retires to the pool.

If there are no workers avaiable in the pool, the WorkerRunnerPool can create additional workers up to the limit specified by the PoolConfig object.

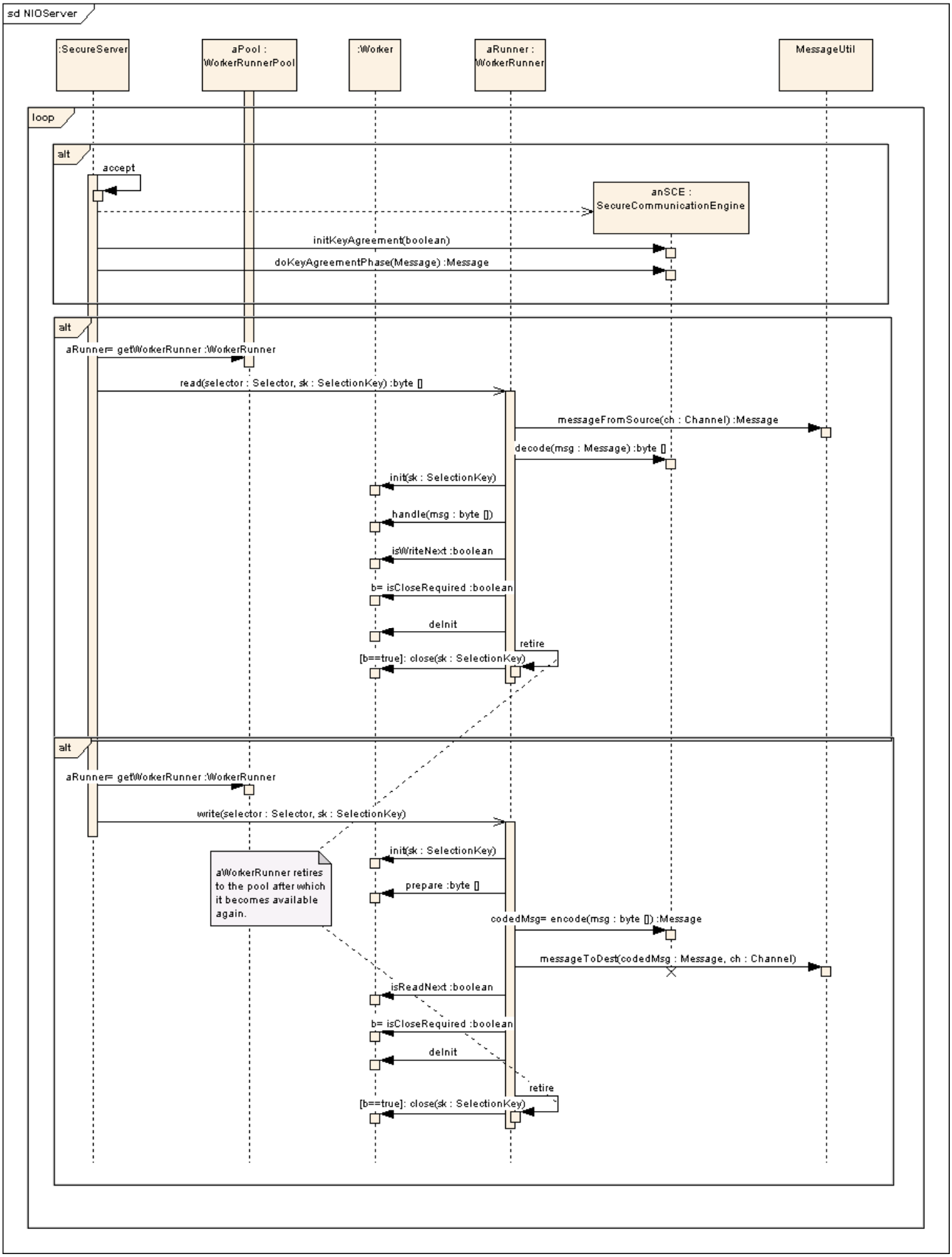

Figure 36: Sending and receiving messages at the server

Implementation view

The class diagram below shows the most important implementation classes and their relationships.

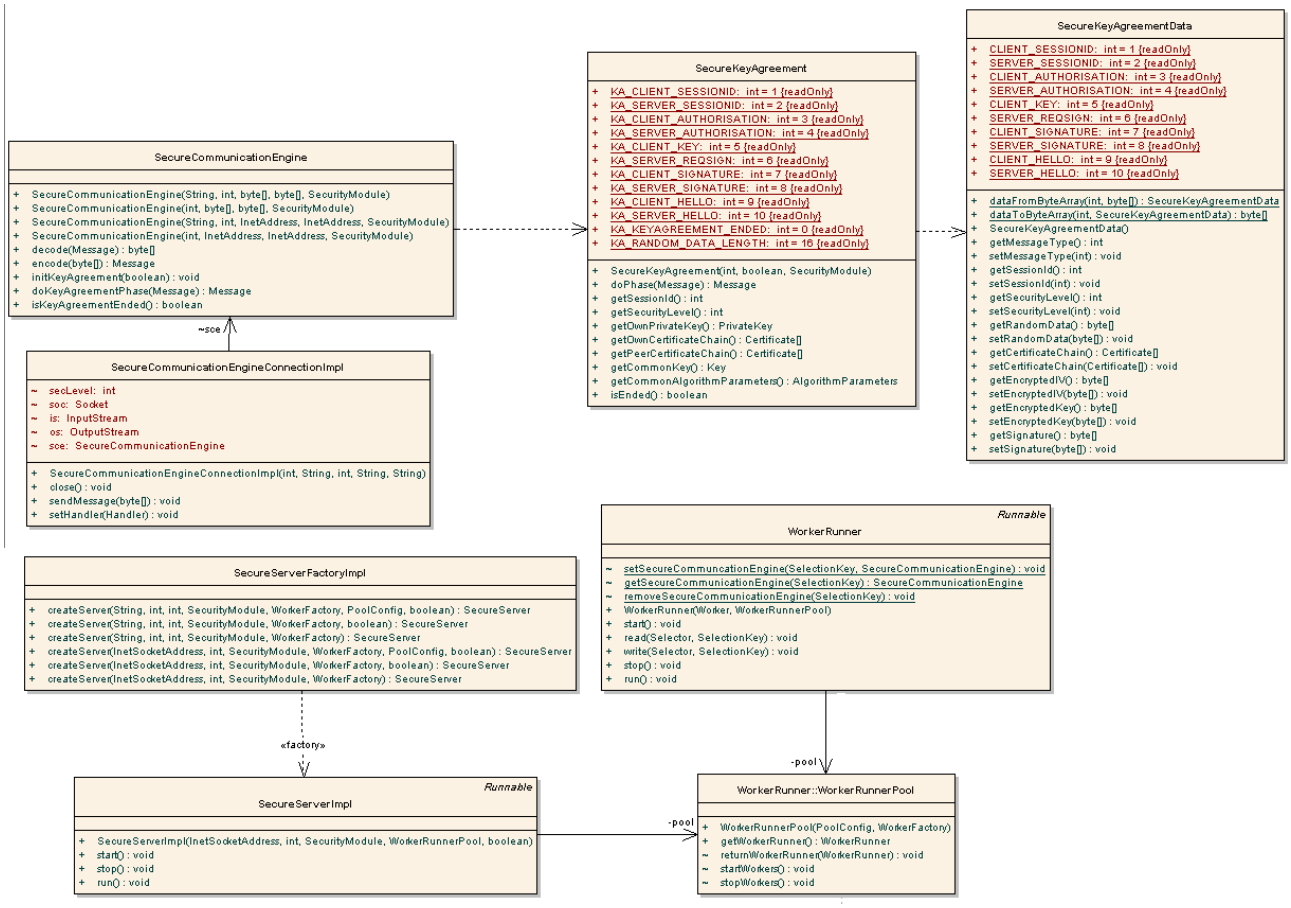

Figure 37 Implementation classes

1.1.1.3. Authentication and Authorization framework

Logical view

The Authentication and Authorization Framework can be divided into the following subsystems:

- CVIS Unit – an in-car, nomadic or road-side unit.
- Host Management Centre (HMC) – operates the back-end infrastructure that is required for distributed authentication and authorization.
- Service Centre – provides the services that CVIS Units can consume.

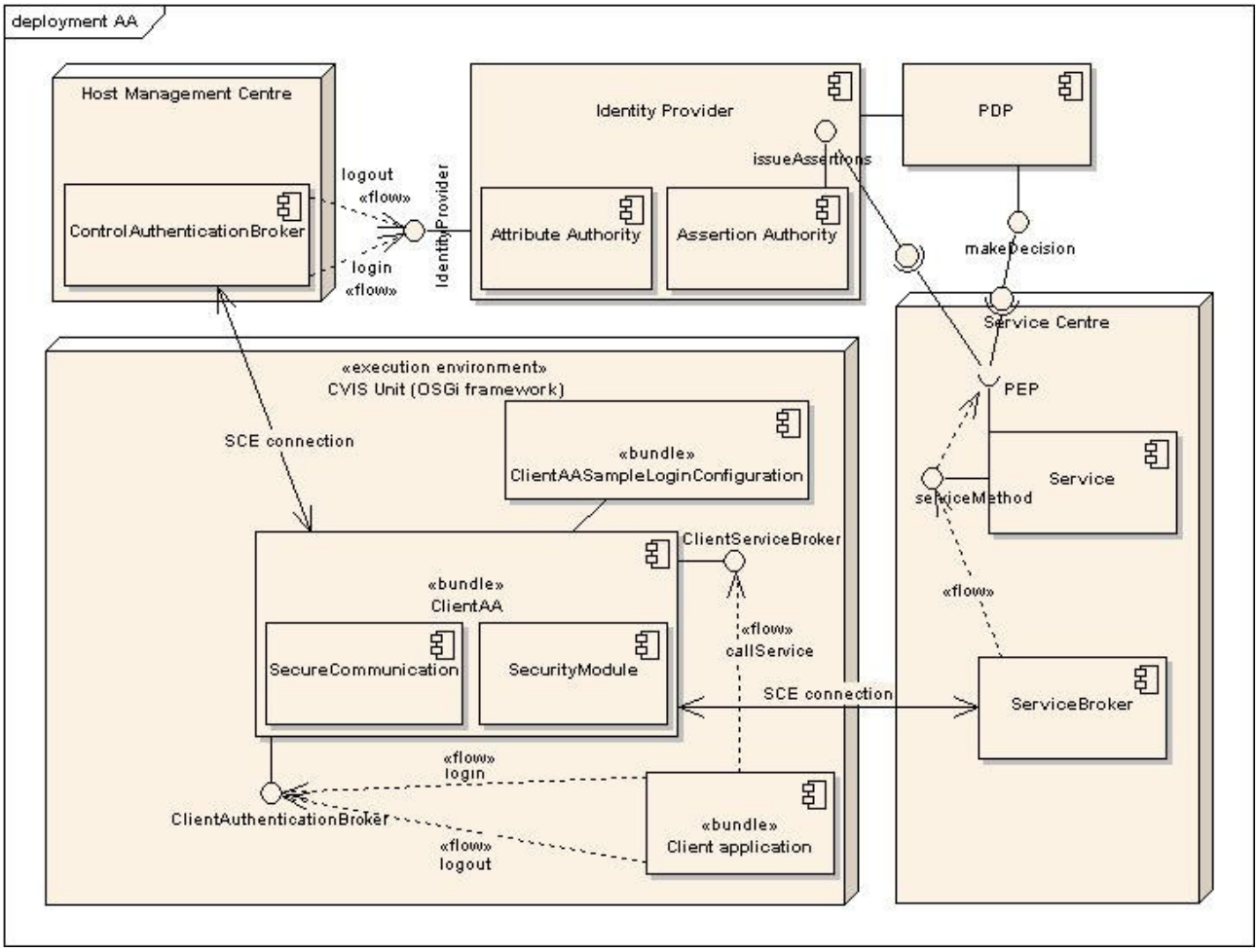

Figure 38: Implementation classes

The above figure gives an overview of the Authentication and Authorization system using a deployment diagram. The following components constitute the AA framework:

- ClientAA bundle (at the CVIS Unit),
- ControlAuthenticationBroker (at the Host Management Centre), and
- ServiceBroker (at the Service Centre).

These components provide infrastructural operations such as Secure Communication between the communicating nodes (CVIS Unit–HMC and CVIS Unit–Service Centre), transferring clients' login/logout requests, and, mainly, transparently delegates service invocations to remote services as if they were local operations. They also ensure that any application-level errors that occurs at the remote node is directed back to the caller.

Service providers need to implement

- the service itself (at the Service Centre) as a Web Service, and
- the client application bundle (at the CVIS Unit)
- Identity providers need to implement
- an identity provider solution,

- a class that implements the IdentityProvider interface, and
- a bundle containing a login configuration (in the picture it is depicted by ClientAASampleLoginConfiguration).

However, the entities Identity Provider, PDP and the Service itself are out of the scope of this documentation.

A CVIS Unit contains the following components:

Client

The person, application etc. that wants to access a service and therefore needs authentication and authorization. It will be referred to as Subject.

ClientAuthenticationBroker

The only entity that deals with any kind of authentication activity at the CVIS Unit. It is responsible for local JAAS login and the initiation of remote login. It enables the client to access the authentication subsystem at the HMC. The ClientAuthenticationBroker and the ControlAuthenticationBroker at the HMC communicate over a SecureCommunication connection. The ClientAuthenticationBroker can also be used to get a Subject to perform JAAS authorization activities on it.

LoginConfiguration

A LoginConfiguration is an entity which is created by the implementer of a Login module. A login configuration defines which login modules need to be used, which callback handlers need to be executed and identifies a security module provider through which implementers may serve their own security module implementation. ClientAuthenticationBroker is backed with a Map of LoginConfigurations where the key is an application name that identifies the associated login configuration.

Client application

Service Providers are required to ship client applications with their services which provide information necessary to access remote services. However, the way in which the form, storage, retrieval etc. of such information is realized is left open to implementers. Client applications call remote services using SOAP messages indirectly, through ClientServiceBroker and ServiceBroker. These brokers communicate with each other through a Secure Connection.

Client applications communicate directly only with the client-side brokers. For local authorization activities they may get a JAAS Subject instance from the ClientAuthenticationBroker, while requests for remote services are handled by the ClientServiceBroker.

ClientServiceBroker

Its role is to check whether the Subject initiating the service call is properly authenticated and if so, it delegates the client's request to the ServiceBroker that resides at the Service Centre. The ClientServiceBroker and the ServiceBroker use a SecureCommunication connection to exchange SOAP messages.

A Host Management Center contains the following components:

ControlAuthenticationBroker

The only entity that deals with any kind of authentication activity at the HMC. It is responsible for remote login at the Identity Provider. On the one hand, it communicates with the ClientAuthenticationBroker through SecureCommunication. Therefore, it hosts a SecureServer instance that handles the communication. Identity Providers Communication between the ControlAuthenticationBroker and the AssertionAuthority takes the form of XML documents, preferably SAML.

Identity Provider

Creates, maintains and manages identity information. Identity Providers should also provide a class (running at the HMC side) that implements an interface (IdentityProvider) which handles login/logout requests coming from the client through ControlAuthenticationBroker. In this way, the responsibility of accessing an Identity Provider and the way this class and Identity Provider communicate each other are outside the scope of the Authentication and Authorization framework.

Apart from that, there might by further entities at the HMC that play important roles in the authorization process. Since, however, they do not communicate directly with any of our interfaces, and are usually obtained from third party providers, their specification is out of the scope of this documentation. For the sake of completeness, these are the Assertion Authority, the Attribute Authority, the PAP, the PIP, the PDP, and the Policy Repository.

A Service Center contains the following components:

Service

An arbitrary security-enabled service, given by a Service Provider.

ServiceBroker

The ServiceBroker's task is to receive the client's request for a service through a SecureCommunication connection, and delegate the information necessary for authorization to the PEP.

PEP

The role of this system entity is to enforce the decision made by the PDP. In other words, it should guarantee that the service it “protects” cannot be reached without proper authorization. Service invocations are intercepted by a filter which plays the role of PEP: it may turn to the Identity Provider to assure that the invoker is logged in and then turns to the PDP to make an authorization decision.

Main interfaces

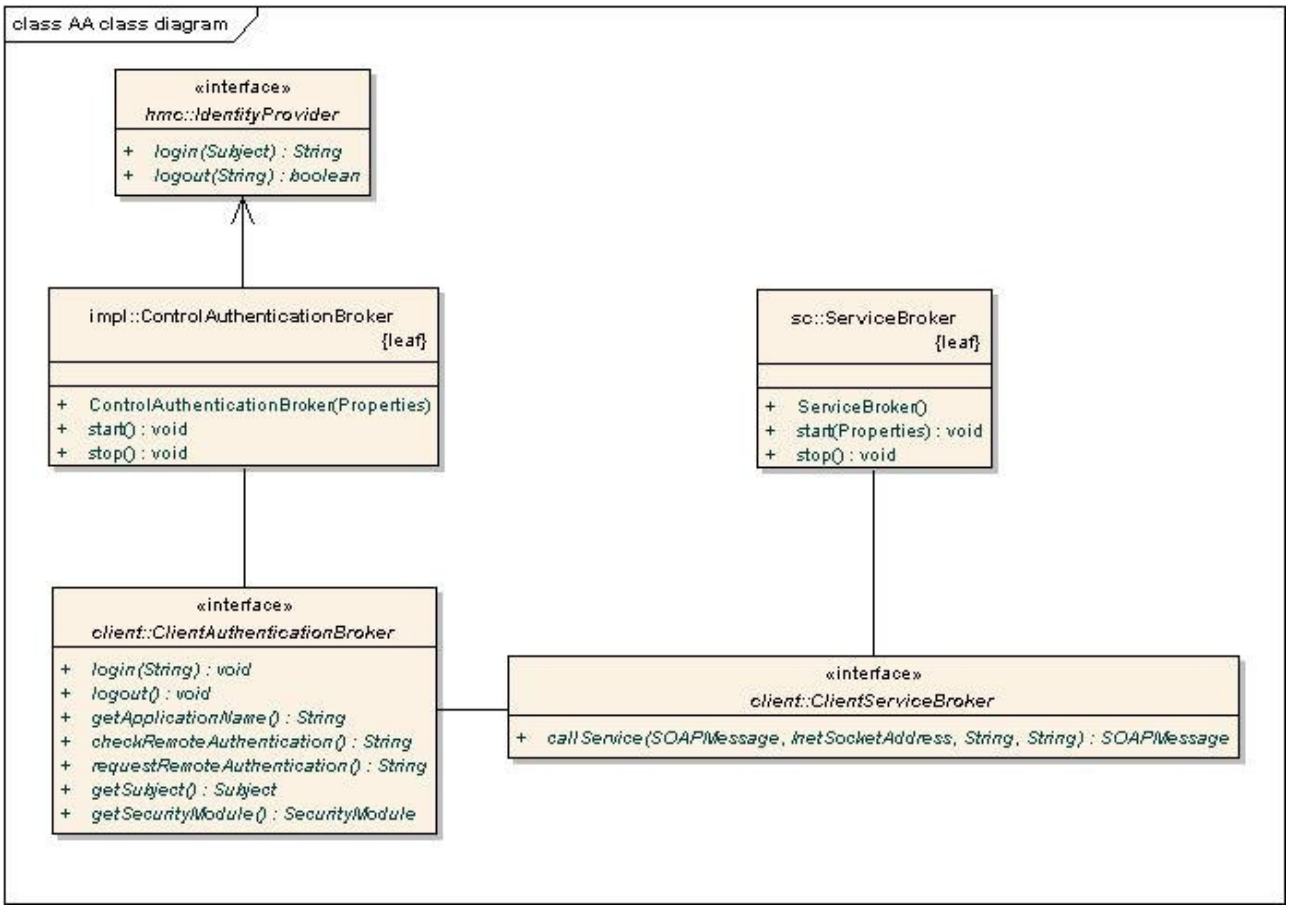

Figure 39 Authentication and Authorization class diagram

In the above figure, ControlAuthenticationBroker and ServiceBroker are classes which provide services (methods) to other brokers only, in the sense that they are not part of the system's external API.

Process view

The following scenarios can occur within the Authentication and Authorization framework:

- Single Sign-On (login)
- Single Sign-Out (logout)
- Service invocation

These are elaborated below.

Authentication (login)

Authentication happens in two phases:

1. Local login – the client is identified locally and may request access to local services. Local authentication and authorization make use of JAAS.
2. Remote login – the client logs in at a HMC. After successful remote login it can request access to remote services.

The Authentication and Authorization framework realizes the Single Sign-On concept: a client has to log in only once, and later its different identities are handled centrally at HMCs. A JAAS Subject represents the client, while the Principals that make up a Subject stand for the various identities of the client.

First local login takes place based on standard JAAS methods. The client that wants to be authenticated initiates the login procedure at the ClientAuthenticationBroker, calling its login method. This method has an application name (a string) as a parameter, which identifies an associated login configuration defining which LoginModules need to be used for authentication.

The ClientAuthenticationBroker instantiates a JAAS LoginContext object, whose login method is responsible for gathering the client's credentials (passwords, keys, biometric data etc.) and creating a Subject instance, which will contain all the identities of the client. After the ClientAuthenticationBroker retrieves the Subject from the LoginContext, it will store it. Local authentication ends at this stage. The only case when local login is not successful is when the client cannot produce the necessary credentials.

The precondition of remote login is obviously that the client be locally authenticated. The ClientAuthenticationBroker starts the remote login procedure by calling the ControlAuthenticationBroker's login(Subject) method, where Subject refers to the newly authenticated client. The Subject is serialized to become transferable through a SecureCommunication connection. As a consequence, LoginModule implementers must take care to use credentials that are serializable, otherwise not all the required information will arrive at the HMC. As a next step, the ControlAuthenticationBroker requests the authorization of the Subject at the Identity Provider. To do this, it a class implementing the interface IdentityProvider needs to be created. This class has the responsibility of knowing both the location of the deployed Identity Manager application and the way how the communication with it should be managed. The Authentication and Authorization framework does not have any constraints on this issue, however, we advice the use of some standard solutions, preferably SAML messages. The IdP then processes the (SAML) request and checks whether the credentials retrieved are appropriate. If there are no problems with the credentials, the IdP authenticates the user by creating an Assertion, an ID of which it sends back as an SAML response to the ControlAuthenticationBroker. Otherwise, it issues an SAML response about the failure meaning the broker will get null value as assertion ID. After this point the assertion ID (or null) is sent back to the client. Finally, the ClientAuthenticationBroker stores the ID with the Subject. A RemoteAuthenticationFailedException signals if null has been received.

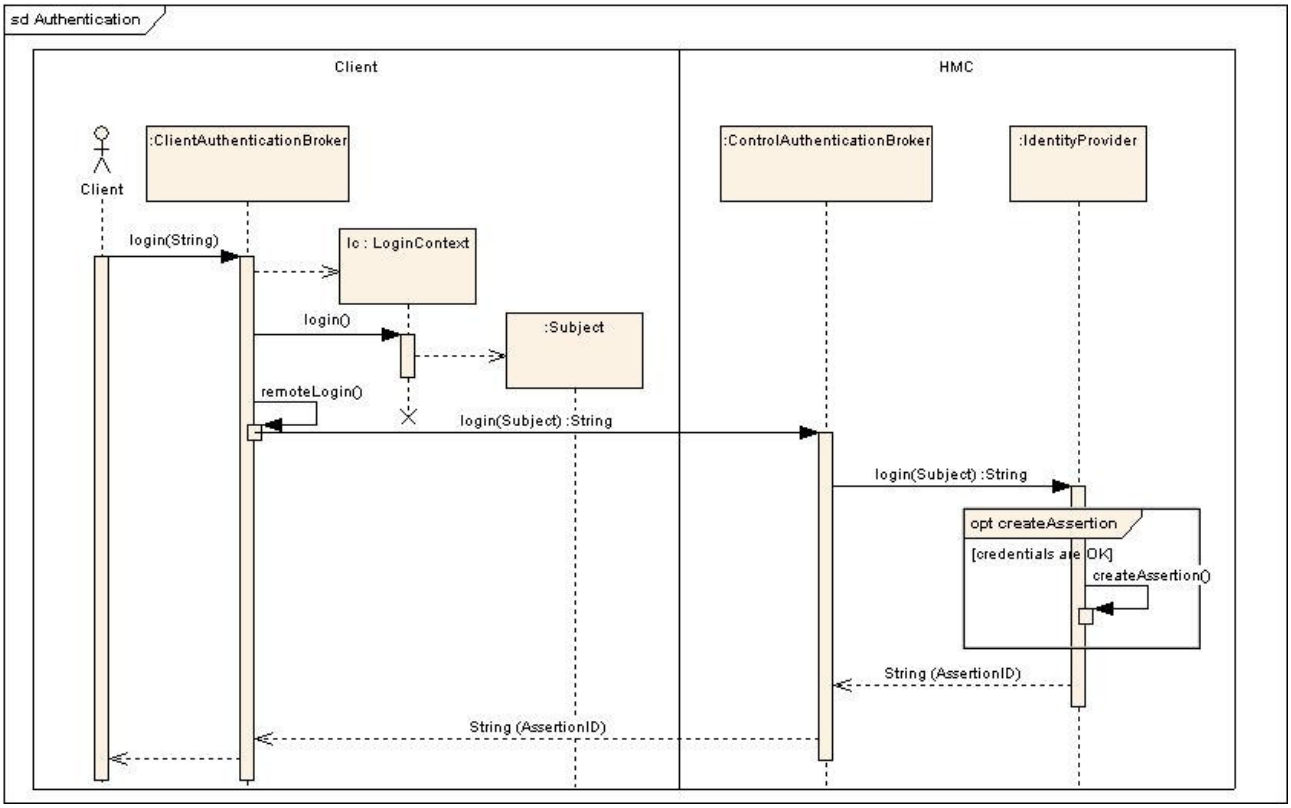

Figure 40 Authentication sequence diagram

Authentication (logout)

The client starts the logout procedure by calling the ClientAuthenticationBroker's logout method. The ClientAuthenticationBroker then retrieves the assertion ID related to the Subject, and initiates its remote logout at the HMC on the basis of the ID. This is done by using SOAP messages. The ControlAuthenticationBroker invokes the logout(String) method of the IdentityProvider interface, where the logic of single sign-out is played with the underlying IdP implementation. This communication is out of the scope of this architecture but usage of SAML is preferred. The last step of the logout is when the ClientAuthenticationBroker removes the Subject from its user information repository.

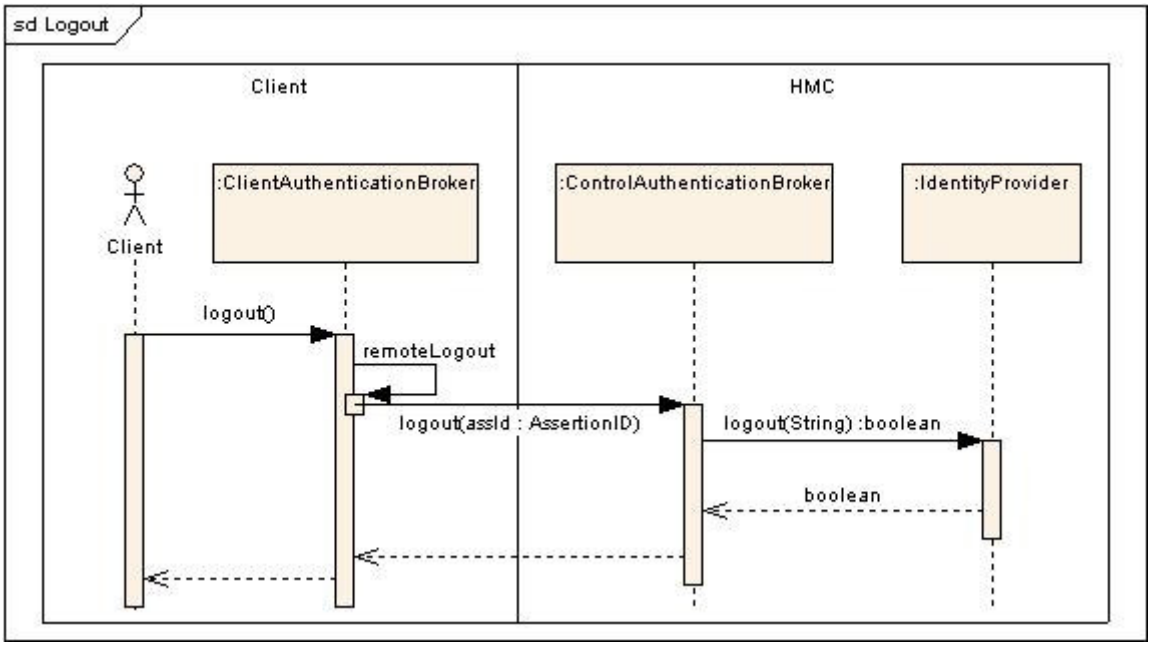

Figure 41 Logout sequence diagram

Service invocation

In order to be able to access a service, the client (Subject) must have the required authorization. Local authorization is completely based on JAAS, therefore the details of this process are out of the scope of this document. When the client wants to access a remote service, it calls the appropriate method of the ServiceProxy that was provided with the service. The ServiceProxy in turn calls the callService method of the ClientServiceBroker, providing it with a SOAP request, the address of the SCE which will handle the message at the Service Centre, and a string describing the location of the service (a URI). Before the ClientServiceBroker delegates the request to the Service Centre, it must check whether the Subject has been authenticated remotely. This procedure involves requesting the remote login of the Subject if no assertion ID is associated with it (i.e.he/she is authenticated only locally). If remote authentication fails, the Subject is not authorized to access the remote service.

In the case of successful remote authentication, the ClientServiceBroker delegates the service request to the ServiceBroker, which resides at the Service Centre. From this point onwards the description is not normative, as the entities in question (PEP, PDP and Service Provider) are out of the scope of the documentation. The ServiceBroker asks the PEP to invoke the service if the Subject has the required privileges to access the resources in question. The PEP orders the PDP to make the authorization decision. If the PDP permits the Subject to access the service, it will be invoked with the given SOAP request, and return the result of the service to the ServiceBroker. If the PDP does not permit access, then the PEP will wrap the negative answer into a SOAP message and return it to the ServiceBroker. The SOAP response is handed back to the ServiceProxy (through ClientServiceBroker).

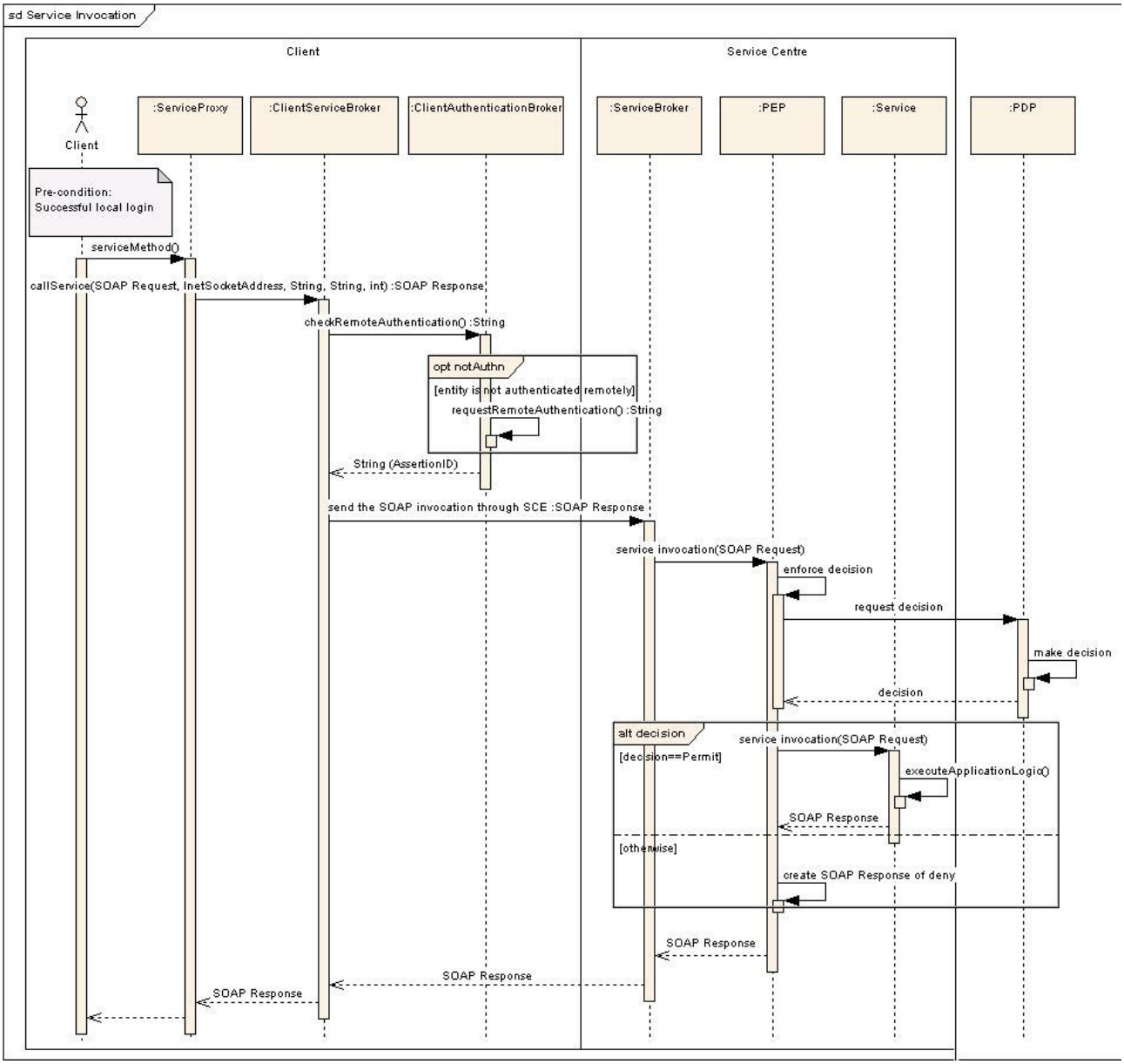

Figure 42 Service invocation sequence diagram

3.4 Broadcast

The "Broadcast" facility is based on the communication functionality which is specified in D.COMM.3.2. FOAM D3.2 describes how this functionality is made available for CVIS applications in JAVA/OSGi.

3.4.1 Overview

The data broadcast facility enables:

a service application to broadcast data,

a service application to subscribe to broadcast data.

The data broadcast service thus consists of two parts, the broadcasting side called, publisher, and the receiving side called subscriber.

3.4.2 Application programming interface

The data broadcast service consists of two main interface classes. DataBroadcastPublisher and DataBroadcastSubscriber. There is also a BroadcastData utility class that encapsulates the broadcast data and its characteristics.

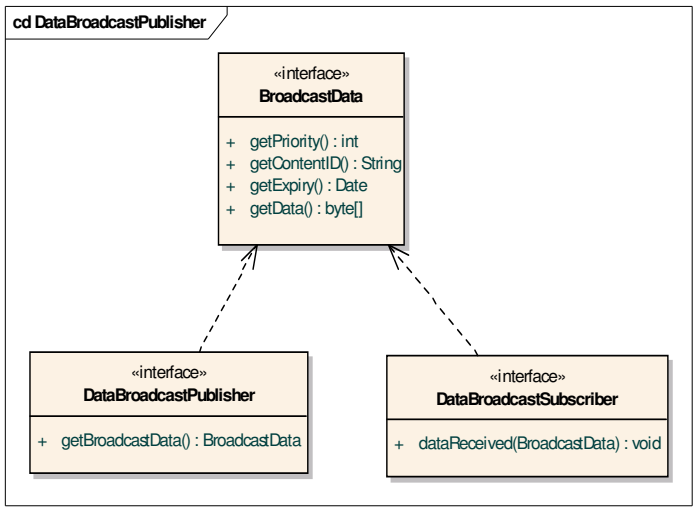

Figure 43: Information model for broadcast facility

The attributes of the BroadcastData, i.e. ContentID, priority, expiry are defined by CALM. Please refer to the COMM 3.2 deliverable for a complete description. This applies to the encoding of the data as well. From a FOAM perspective the data is regarded as an arbitrary array of bytes.

3.4.3 Information model

The broadcasted data is always identified by a ContentID as defined in the CALM standard for broadcast data. The broadcasted data has also an expiry and priority associated with it. Expiry may be time and/or location, i.e. the broadcasted data is only valid in a certain region.

3.4.4 Interaction model

The sequence diagram below gives an overview of the entire flow of data. To the far right is an application that desires to broadcast data and to the far left is another application that would like to receive the same data by registering as a data broadcast subscriber.

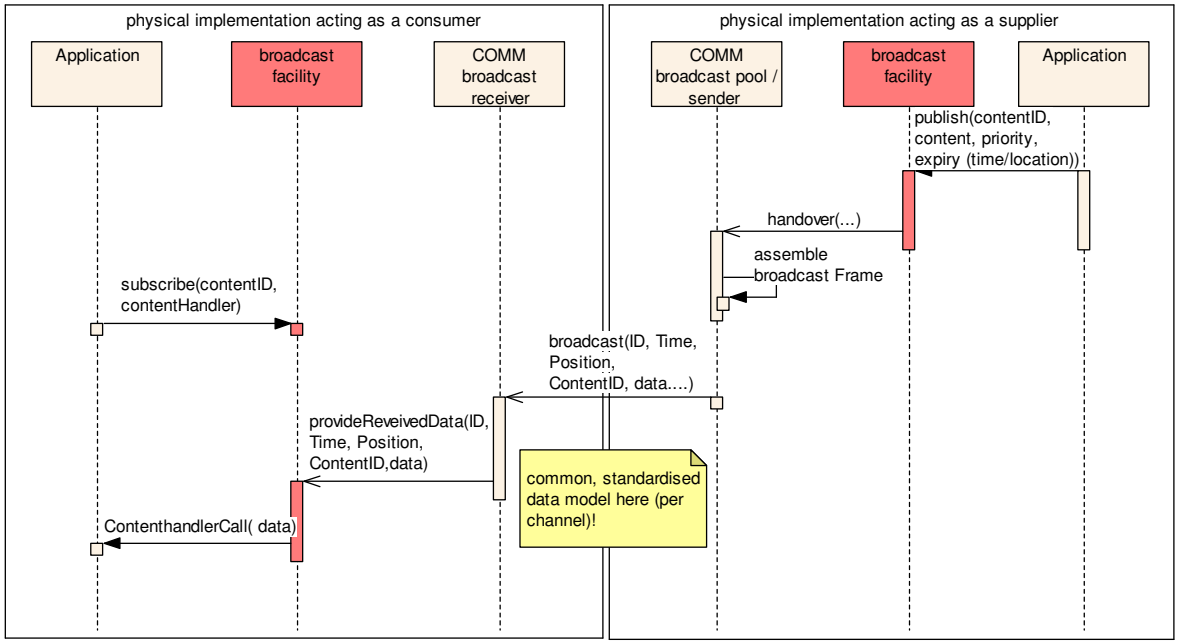

Figure 44: Interaction for broadcast data

3.4.5 High level composite architecture

Figure 44 identifies one main architectural component of the facility towards applications: The broadcast facility itself.

3.5 Connection manager

The connection manager API will provide access to CALM functionality through an 'open' call, in which the quality of the requested service can be defined using a set of parameters. Further details can be found in D.FOAM.3.2 chapter 8.

The connection manager may automatically choose the connection in order to use the most efficient communication mechanism to transport the messages from the sender to the receiver.

Choosing between available connection factories can be done automatically in the connector based on service properties depending on the chosen policy.

3.5.1 Overview

The IO connector service from the OSGi specification matches the connection manager requirements very well.

The sub-projects may use the standard connection factories defined by the OSGi specification for their communication needs and are encouraged to provide connection factory services where specific protocols are needed.

3.5.2 Application programming interface

Javax.microedition.io.Connection

Type: public abstract «interface» Interface

Package: connection manager

An existing standard interface:

It is extended by HttpConnection, DatagramConnection, InputConnection, OutputConnection etc.

Javax.microedition.io.Connection Interfaces

Method	Type	Notes
close ()	public: void	

org.osgi.service.io.ConnectionFactory

Type: public abstract «interface» Interface

Package: connection manager

An existing interface:

See J2ME connector and OSGi Release 4 ConnectionFactory

org.osgi.service.io.ConnectionFactory Interfaces

Method	Type	Notes
close ()	public: void	
getScheme ()	public: String	
createConnection (string, int, boolean, Dictionary)	public: Javax.microedition.io.Connection	param: uri [string - in] param: mode [int - in] param: timeouts [boolean - in] param: options [Dictionary - in]
open ()	public: void	MUST be called by the implementation of the connector service before any create-Connection method is called. It MAY be used to create resources that are need for the correct operation of this package.

org.osgi.service.io.ConnectorService

Type: public abstract «interface» Interface

Package: connection manager

Existing interface from OSGi R4:

org.osgi.service.io.ConnectorService Interfaces

Method	Type	Notes
open (String)	public: Javax.microedition.io.Connection	param: address [String - in]
open (int, String)	public: Javax.microedition.io.Connection	param: mode [int - in] READ, WRITE, READ_WRITE param: address [String - in] The address of the destination, in URI format
open (boolean, int, String)	public: Javax.microedition.io.Connection	param: timeout [boolean - in] param: mode [int - in] READ, WRITE, READ_WRITE param: address [String - in] The address of the connection, in URI format
openDataInputStream (String)	public: Java.io.DataInputStream	param: address [String - in] The address of the destination, in URI format.
openDataOutputStream (String)	public: Java.io.DataOutputStream	param: address [String - in] The address of the destination, in URI format.
openInputStream (String)	public: Java.io.InputStream	param: address [String - in] The address of the destination, in URI format.
openOutputStream (String)	public: Java.io.OutputStream	param: address [String - in] The address of the destination, in URI format.

org.cvisproject.comm

Type: public abstract «interface» Interface

Extends: Connection.

Package: calm

This interface provides methods for creating the objects needed for a communication utilizing the CALM technology.

CALMConnection Interfaces

Method	Type	Notes
getParameters	public: Dictionary	Returns the QoS parameters associated with this connection.
setParameters	void	param: dict [Dictionary - in]

3.5.3 Information model

Standard IO connector service

The following figure shows a class diagram for the standard connection manager module. The connector service is the central place where an application can obtain a connection. This connector service must be provided by the OSGi framework or by a system application running on the OSGi framework.

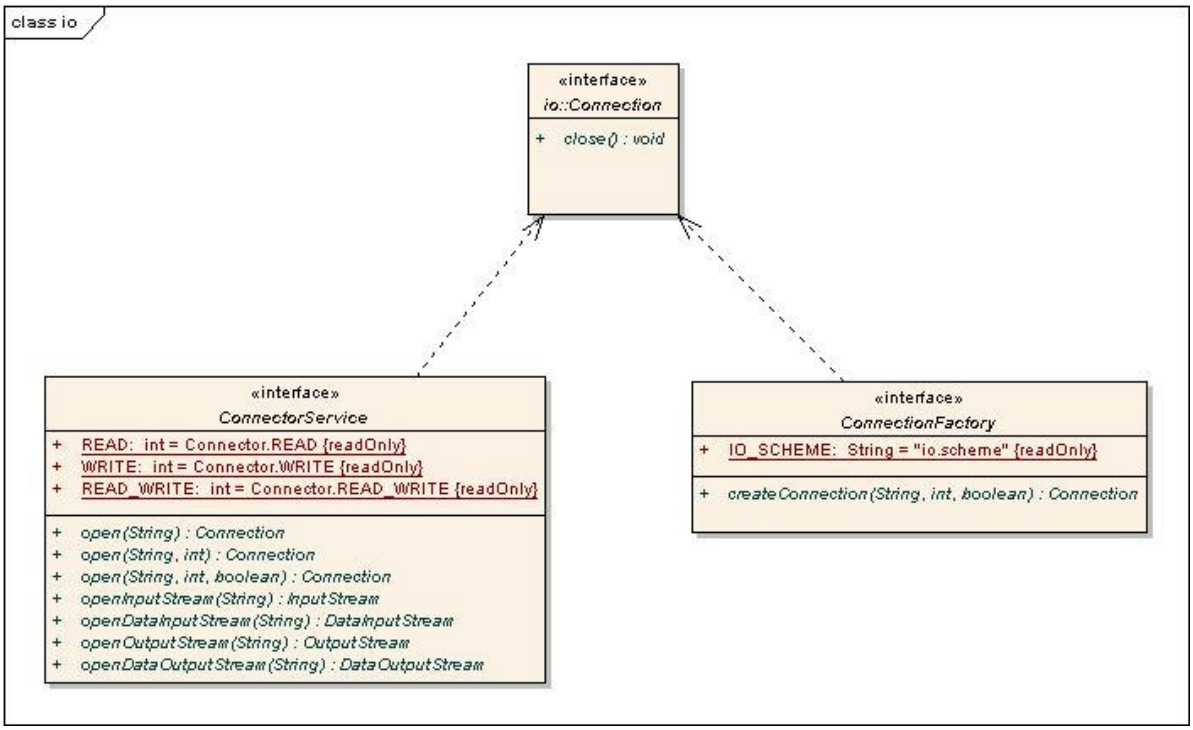

Figure 45: Class diagram for connection manager

SOAP connection factory

The SOAP connection factory extension of the connection manager is an interface that adds priorities management and control over the connectivity to Internet using the different adapters - GPRS, WiFi etc. based on these priorities.

The extension consists of two parts: an interface for SOAP conformant with the ConnectionFactory interfaces described in the previous chapter; and a separate interface for an AdapterManager that is responsible for the management of network adapters and modems.

The SOAP factory interface is a thin abstraction layer that allows the creation of a SOAP message and its sending. No queuing will be done in the soapconnector - if the message cannot be sent then it must be returned synchronously to the calling application and it may have a queue inside to save it and try again later.

In general the SOAP connection factory implementation provides the functionality to send messages, block low-priority messages if a high-priority message has to be sent, and request connecting over a fixed network adapter as opposed to the automatic connecting that the operating system does when you simply open a socket.

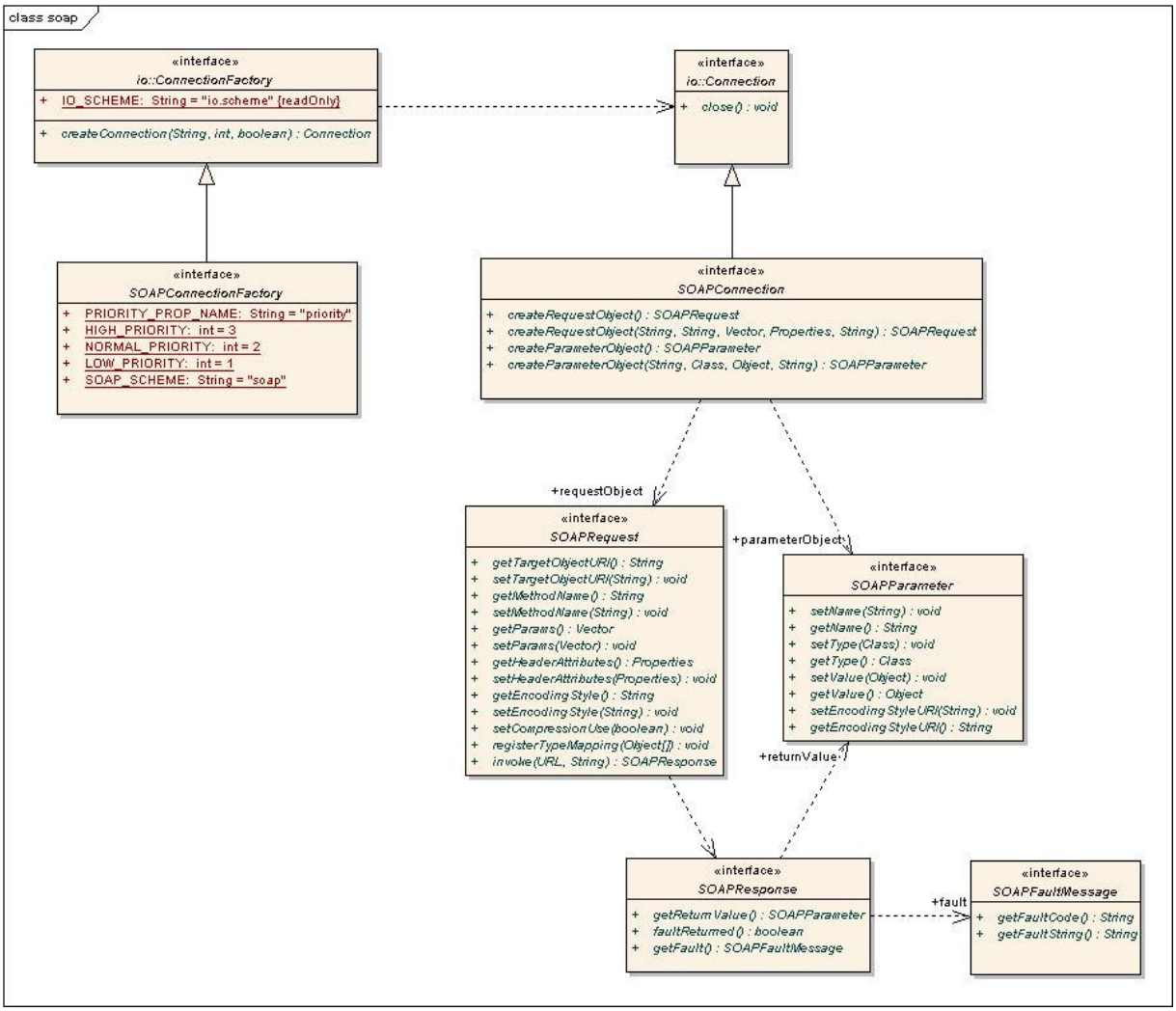

Figure 46: A SOAP extension of the ConnectionFactory

3.5.4 Interaction model

The following sequence diagram shows the steps when an application needs to open a connection:

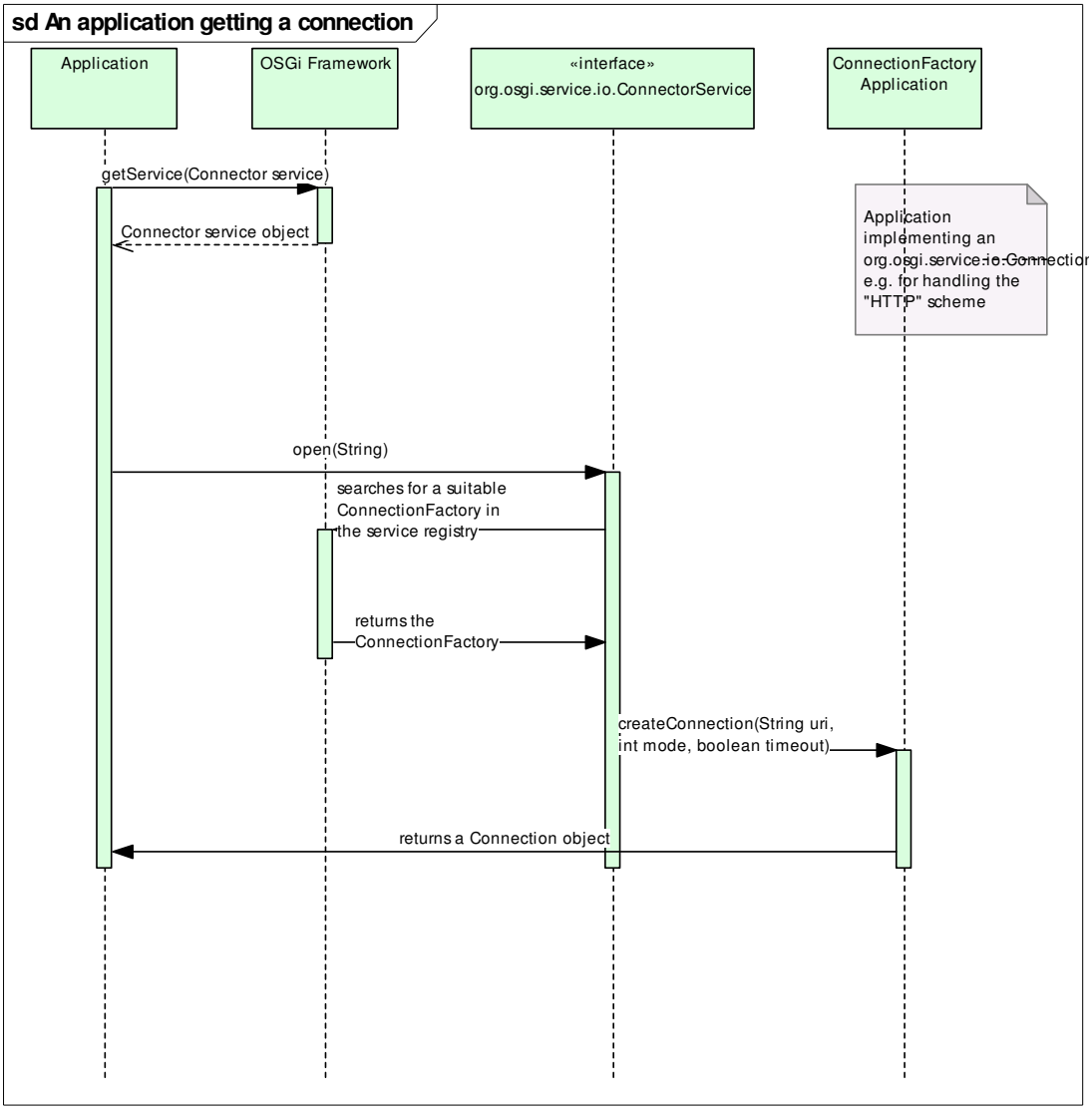

Figure 47 Application getting a connection

Getting a connection is done in two steps: first the application acquires the connector service from the service registry, and then asks the connector service for a connection object. Once a connection object is obtained, its methods will be used for sending messages during the lifecycle of the application.

Sending a SOAP message

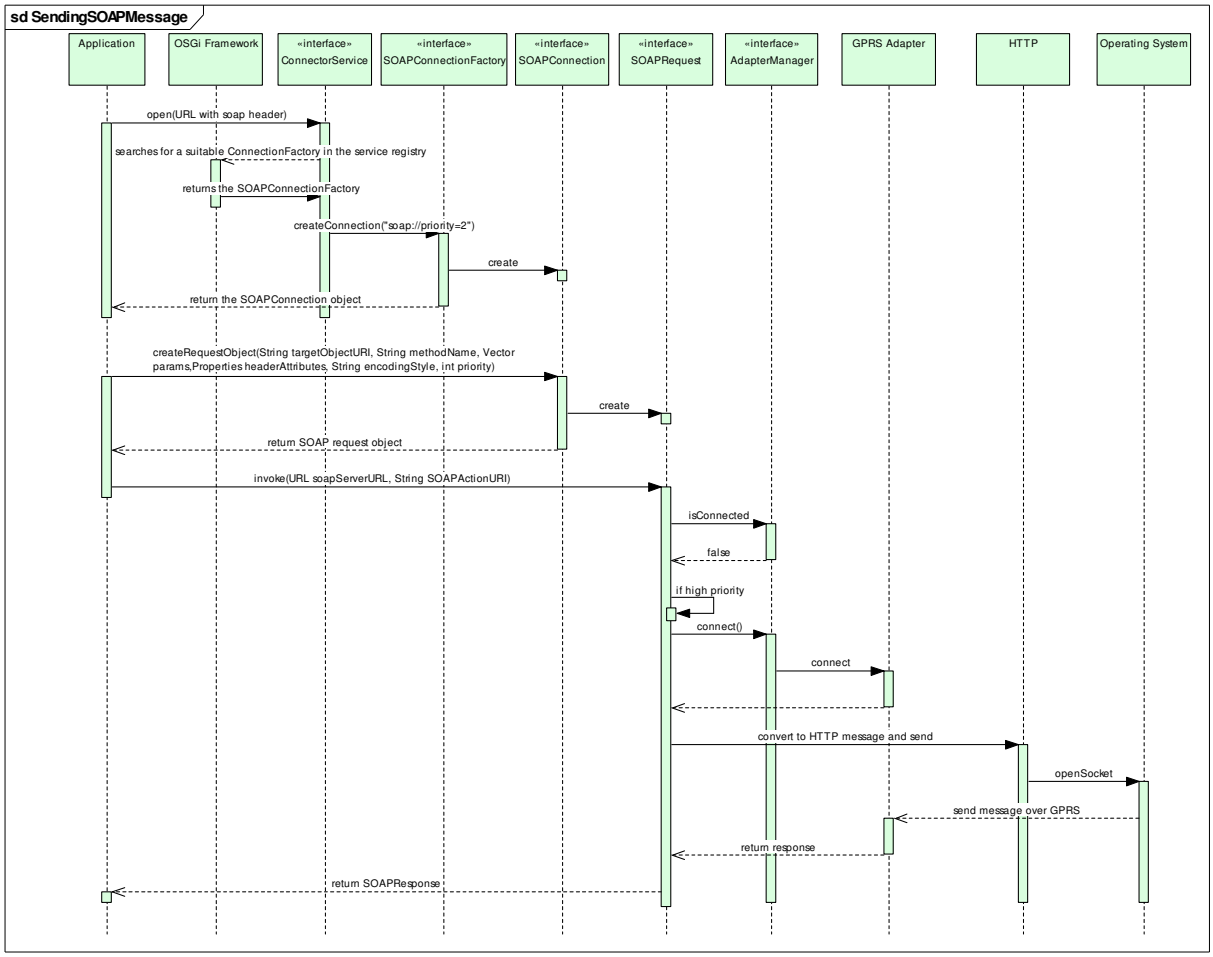

Figure 48: Sending a SOAP message

3.5.5 High level composite architecture

See chapter in "Information Model". Components such as *ConnectionFactory* and *ConnectorService* are included in the descriptions.

3.6 Human machine interface

The CVIS project will not by itself develop a reference HMI architecture for this purpose, nor will it develop a reference implementation. The reason for this is that such architecture is developed by the "AdaptIve Driver-vehicle interfacE" (AIDE)" integrated project [AIDE-1], a project in which all major vehicle OEMs in Europe are participating. An overview of this architecture is contained in Annex 4. This overview serves as a guideline for deployment of CVIS units in the real world.

However, for the CVIS project, FOAM will provide a non-normative application manager that will provide a basic default HMI to the end user. It is based on Java AWT and is completely skin-able. This shall thus not be seen as HMI reference architecture and is not considered part of the CVIS specification as such.

3.6.1 Overview

In this HMI chapter an example for the usage of the HMI is described.

The following sections show how an application manager facility is implemented.

3.6.2 Application programming interface

The API is provided through JAVA "Abstract Windowing Toolkit" (AWT).

3.6.3 Information model

Here the information model of the example facility - the application manager is described.

The service application shall use the ViewListener interface to inform the ApplicationView on updates and to request focus.

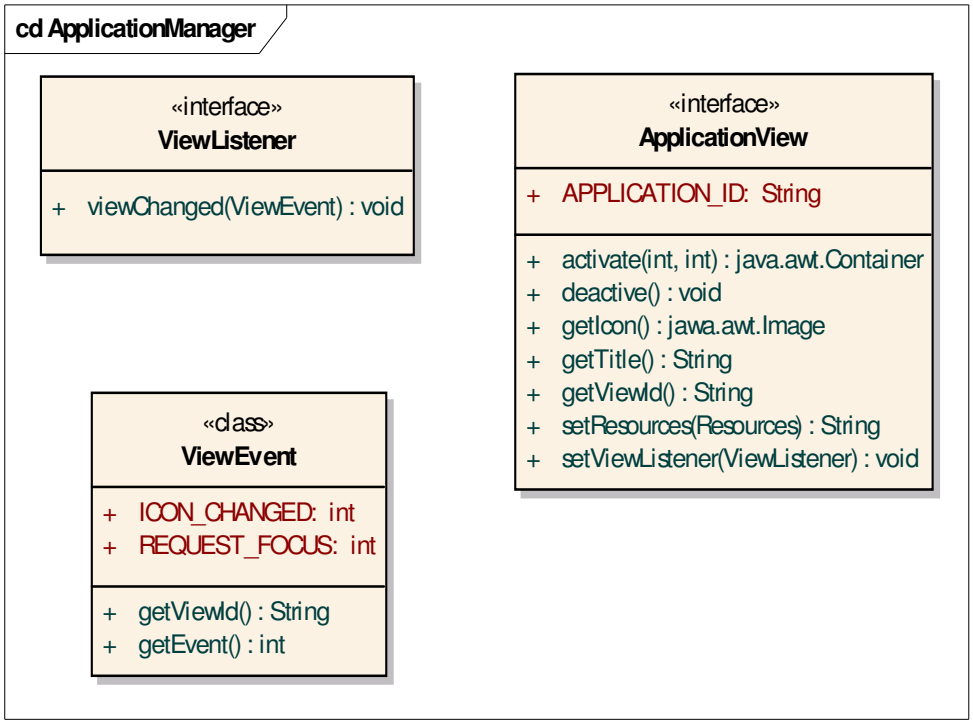

Figure 49: Application manager - example information model

3.6.4 High level composite architecture

The picture below shows the application manager inside a CVIS host. The HMI in itself is not considered part of the service platform. It uses the management agent for end-user driven provisioning.

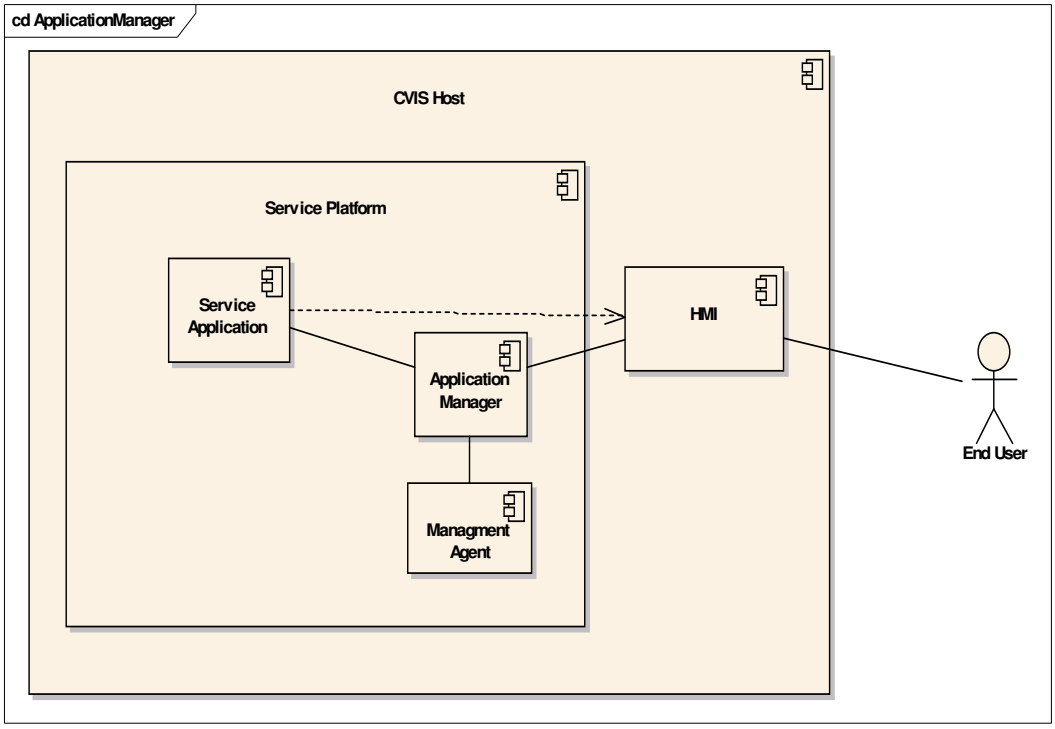

Figure 50: HMI high level composite architecture

3.7 Local device tree

The primary objective of the "Local Device Tree" (LDT) is to provide access to device related status information like (in the case of a vehicle) vehicle speed, fog lamp status, GPS position etc.

In addition the LDT will allow:

To manage the device database (discover, expand or delete database elements) locally.

Optional: To manage or access the device database from a remote server by using the OMA device management specification.

The local device tree is described in detail in D.FOAM.3.2 chapter 13.

3.7.1 Overview

The use cases identified for the LDT API are:

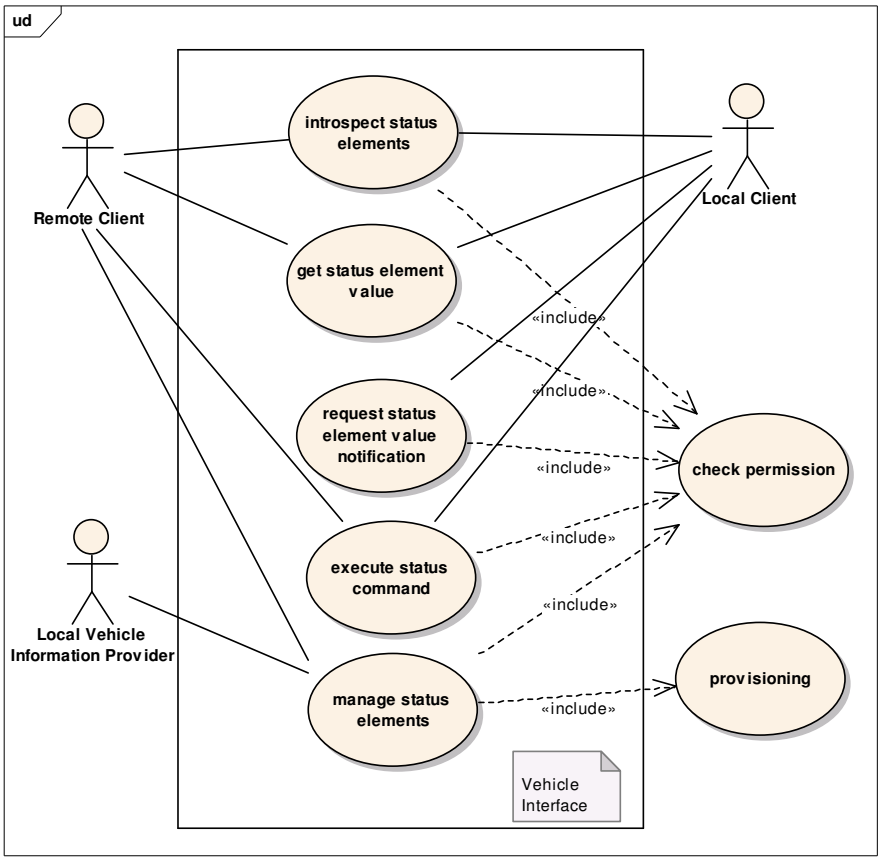

Figure 51: Device tree use cases

3.7.2 Application programming interface

The API for the LDT is specified in Figure 52. Be aware of the encapsulation of data in cell-objects, shown in Figure 63.

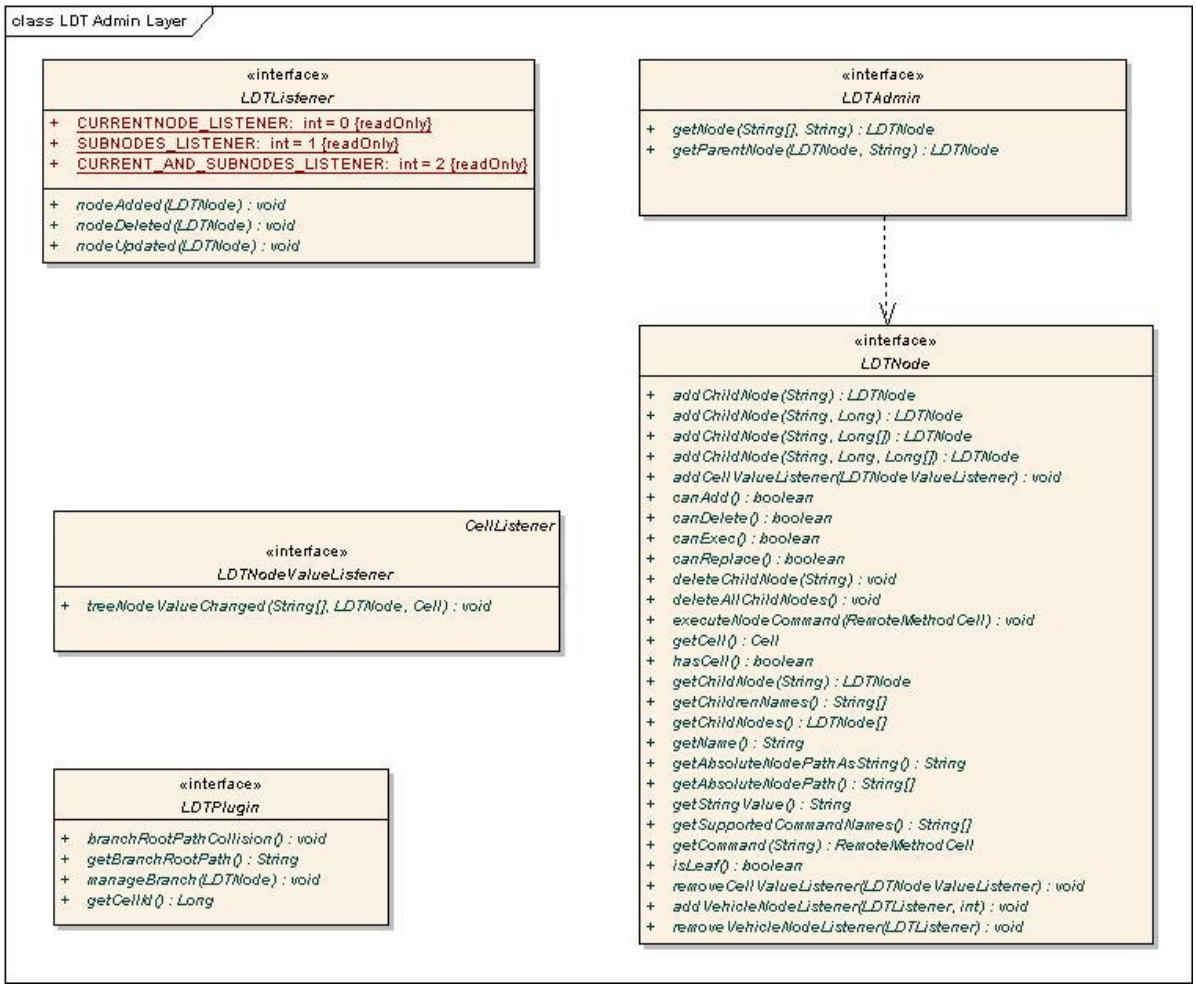

Figure 52: Class diagram LDT admin layer

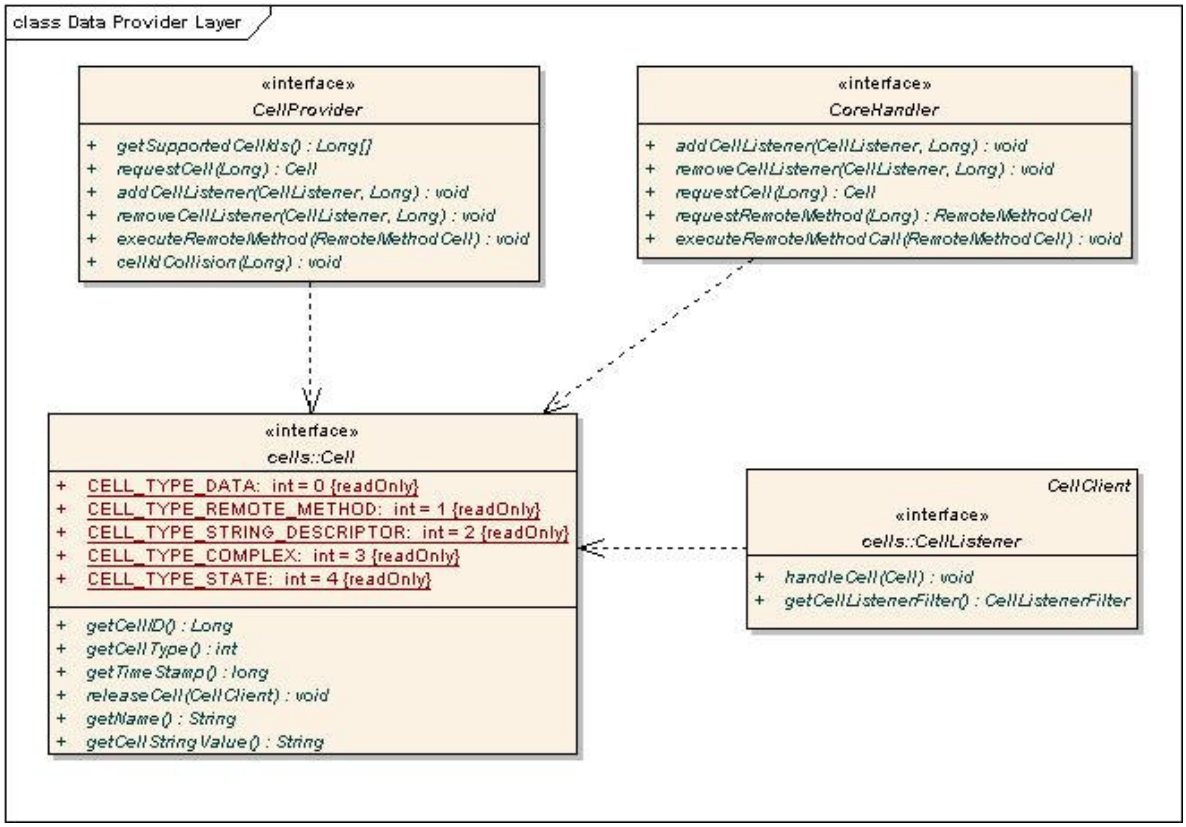

Figure 53: Class diagram data provider layer

3.7.3 Information model

Main elements are:

A cell is an abstract basic class for providing metadata about a property value..

A data cell contains the physical value of the property data.

The complex cell is a container for data cells, i.e. each of the elements part of the complex data, is represented by a cell.

String descriptor represents a string status value. Such a status value could be the VIN, or registration number of a vehicle.

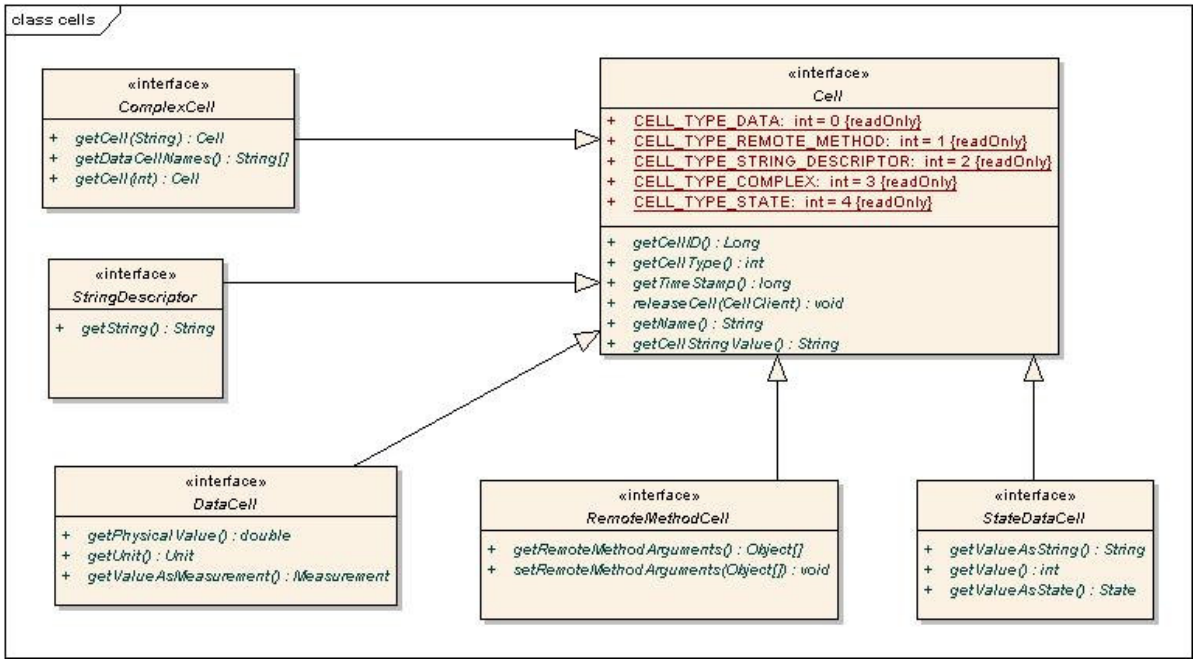

Figure 54: Local device tree information model

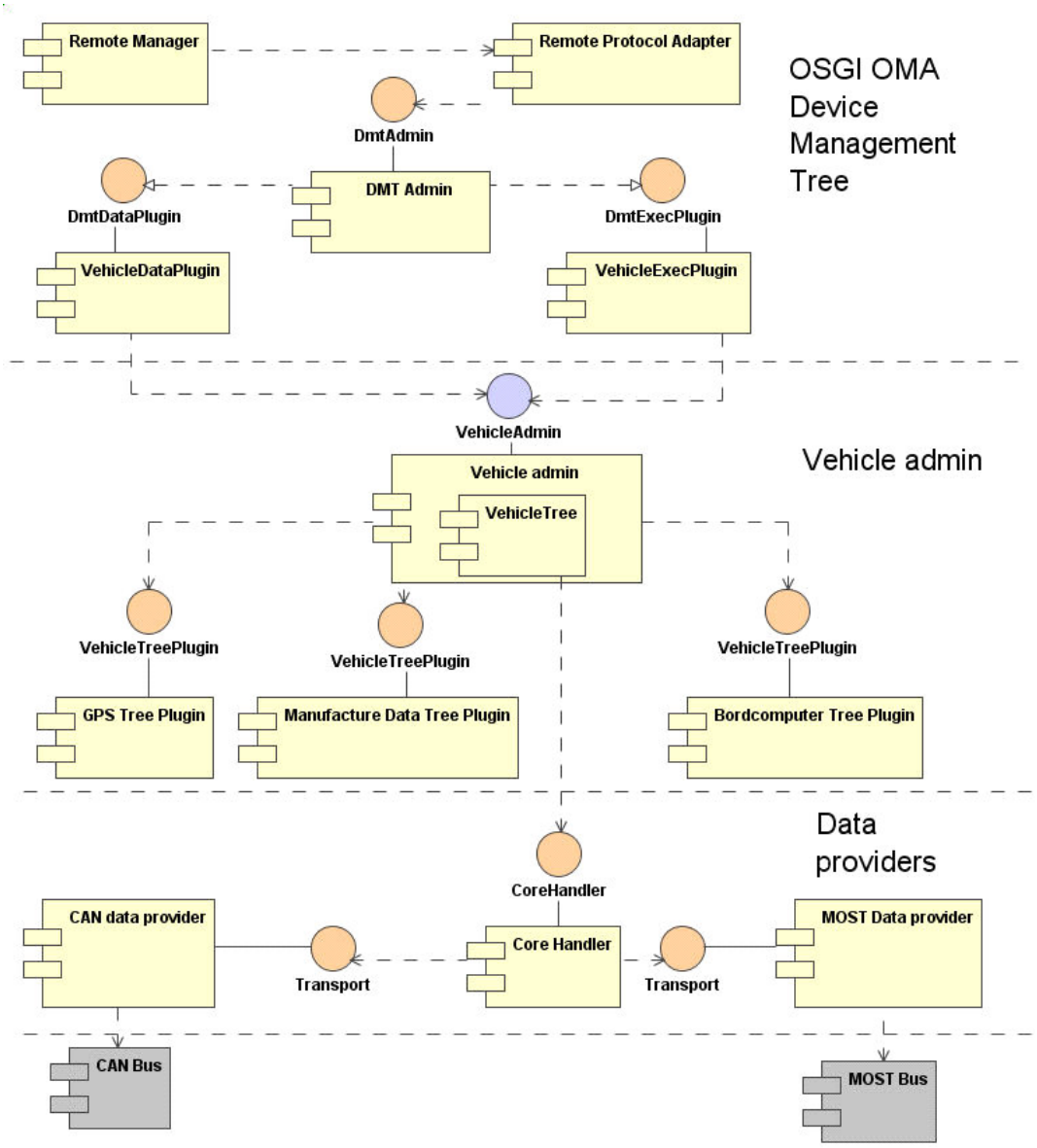

Figure 55: Entities for the interface to device sensors

4 Domain facilities

This section describes the set of domain facilities provided in CVIS. While the basic facilities described in the previous section features basic execution management and communication, the domain facilities provide common ITS domain services. The domain facilities may be available in a CVIS host. The availability is dependent of its relevance, since domain facilities are useful for some applications but not for all. Domain facilities can be downloaded to be available on demand. The following domain facilities are specified:

Position and map matching; which provides position information that may be provided as a map position (map matching)

Infrastructure position; which provides the position of CVIS nodes by localizing them using the wireless sensor network

Map provision; which supply maps and updates of maps

Location reference; which provides encoding and decoding of Agora-C location references

Geo-spatial platform (GSP); which provides GIS functionality such as "Geo-code", i.e. to convert an address to X,Y coordinates, "MapDisplay & Route", i.e. to calculate the optimal route from a single location to one or several destinations. GSP also facilitate provision of traffic data to the GSP from external sources. This traffic data may then be used to enhance the response provided by "MapDisplay & Route" requests

Cooperative traffic information; which provides access to traffic status information which is determined in a cooperative way.

Billing and payment; which provides common facilities for billing and payment. The billing and payment facilities are completely covered by the GST S-PAY [GST.S-PAY.3.2], and are not described any further in this document.

Each of the domain facilities is presented in the next sub-sections applying the following viewpoints:

Overview*;* which provides an overall introduction to the facility

Application programming interface*;* which describes the API accessible for the users of the facility (typically applications, but a facility may also be used by other facilities)

Information model*;* which specifies the facility from an information perspective describing information objects of the facility domain.

Interaction model*;* which specifies main usage scenarios associated with the facility.

High level composite architecture*;* which specifies the main components constituting the facility (this perspective is optional, since some facilities consists of only one main component).

This document (D.CVIS.3.4) includes specifications of interest for the users of the facilities. Further details as well as the internal design are specified in the corresponding D.SP.3.2 documents.

4.1 Position and map matching

This sub-section is based on the D.POMA.3.2 specification document. In this document the focus is on the external interfaces. For discussions of the internal architecture of the position and map matching we refer to the D.POMA.3.2 specification document.

4.1.1 Overview

The position and map matching facility provides position information, the position may be provided as a map position (map matching). The use case model of the position and map matching facility is specified in Figure 56

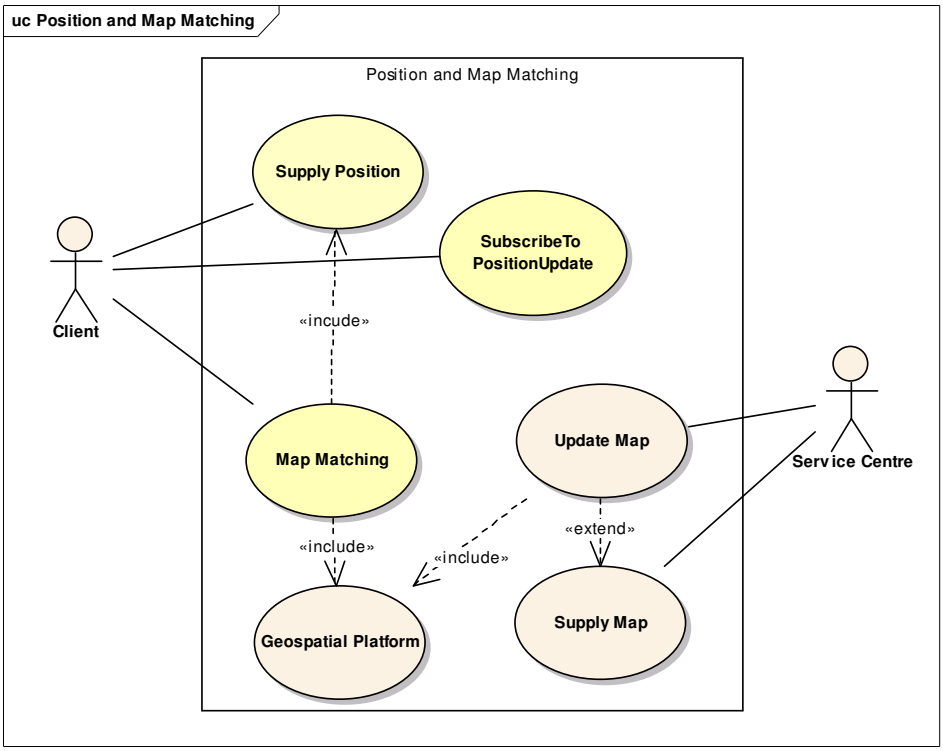

Figure 56: Position and map matching use cases

The main use case is the "Supply Position" which provides position information, the map matching which uses the "Supply Position" and the "Geo-spatial Platform" (GSP) to provide position on a map and the "Subscribe to Position Update", which enables to a client application to be notified on position updates. The service centre is responsible for supplying and updating maps.

Sensors based positions and map-matched positions will be computed each second. The sensors position interface and map-matched position interface will provide an interface to access these positions in the following ways:

A client application requests a single position (either map-matched or not) and immediately gets an answer.

A client application requests a number of last positions (either map-matched or not) and immediately gets an answer

A client application registers and asynchronously receives the new positions as soon as

they are computed.

4.1.2 Application programming interface

The interfaces of the position and map matching facility are specified in Figure 57.

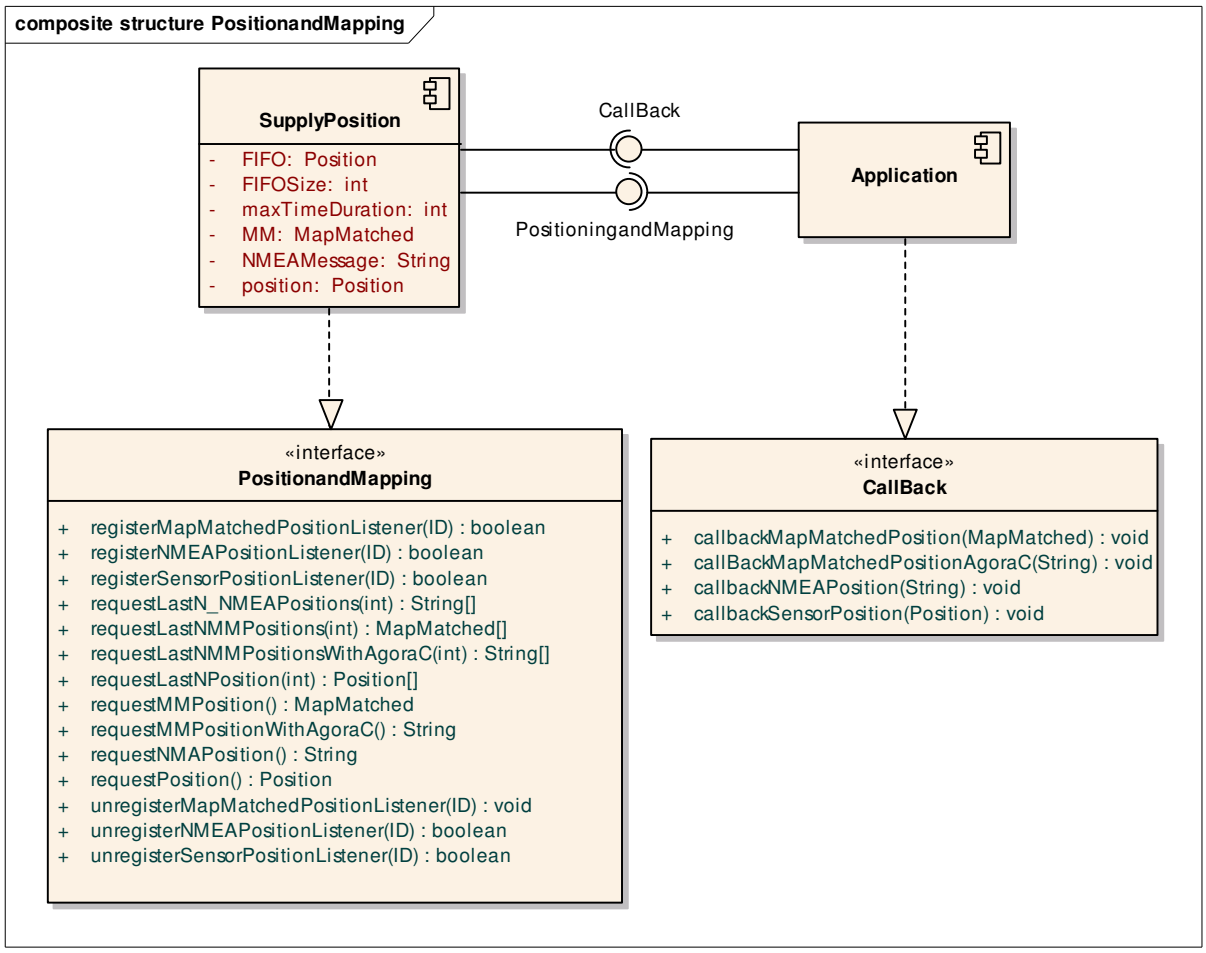

Figure 57: Interfaces of the position and map matching facility

The SupplyPosition component provides the PositionandMapping interface to the client applications. The client application needs to implement the Call back interface to be able to register for notifications of position updates.

The following table presents the methods available on the SupplyPosition Interface.

Method	Parameters	Returns	Description
requestPosition	None	Non map-matched struct	Non-blocking Returns the last known position.

Method	Parameters	Returns	Description
requestLastNPositions	Int N : number of last positions required	List of non map-matched struct.	Non-blocking Returns a list containing the N last known positions. In case N is larger that the number of buffered position, the list size will be lower than N.
requestNMEAPosition	None	String : NMEA message	Non-blocking Returns the last known position encoded in an NMEA message
requestLastN_NMEAPositions	Int N : number of last positions required	List of NMEA messages strings	Non-blocking Returns a list containing the N last known positions encoded in NMEA strings In case N is larger that the number of buffered position, the list size will be lower than N.
requestMMPosition	None	Map-matched position struct	Non-blocking. Returns the last known map-matched position
requestLastNMMPositions	Int N : number of last positions required	List of map-matched position structs	Non-blocking Returns a list containing the N last known map-matched positions. In case N is larger that the number of buffered position, the list size will be lower than N.
requestMMPositionWithAgoraC	None	Map-matched position struct, String	Non-blocking. Returns the last known map-matched position and a location reference

Method	Parameters	Returns	Description
requestLastNMMPositions WithAgoraC	Int N : number of last positions required	List of map-matched position structs, String	Non-blocking Returns a list containing the N last known map-matched positions. In case N is larger that the number of buffered position, the list size will be lower than N and a location reference.
RegisterSensorPositionListener	Pointer to the CallbackSensorPosition interface	Boolean : succeeded	Allows an application to register its callback method in order to asynchronously receive the sensors position structs (see the CallbackSensorPosition interface)
UnregisterSensorPositionListener	Pointer to the CallbackSensorPosition interface	None	Tells the POMA service not to send positions anymore via the callback.
RegisterNMEAPositionListener	Pointer to the CallbackNMEAPosition interface	Boolean : succeeded	Allows an application to register its callback method in order to asynchronously receive the NMEA position string (see the CallbackNMEAPosition interface)
UnregisterNMEAPositionListener	Pointer to the CallbackNMEAPosition interface	None	Tells the POMA service not to send positions anymore via the callback.
RegisterMapMatchedPositionListener	Pointer to the CallbackNMEAPosition interface	Boolean : succeeded	Allows an application to register its callback method in order to asynchronously receive the map-matched position struct (see the CallbackMapMatchedPosition interface)
UnregisterMapMatchedPositionListener	Pointer to the CallbackMapMatchedPosition interface	None	Tells the POMA service not to send positions anymore via the callback.

The following interface shall be implemented by any application which requires asynchronous reception of position data.

Method	Parameters	Returns	Description
CallbackSensorPosition	Non map-matched struct	Void	Will be called asynchronously by the POMA service when an application registered this interface.
CallbackNMEAPosition	String : NMEA message	Void	Will be called asynchronously by the POMA service when an application registered this interface.
CallbackMapMatchedPosition	Map-matched struct	Void	Will be called asynchronously by the POMA service when an application registered this interface.

4.1.3 Information model

The basis position information used in the positioning and map matching facility is specified in Figure 58. See also the D.POMA.3.2 architecture specification document for more details of the data format for different encodings, e.g. NMEA and Agora-C.

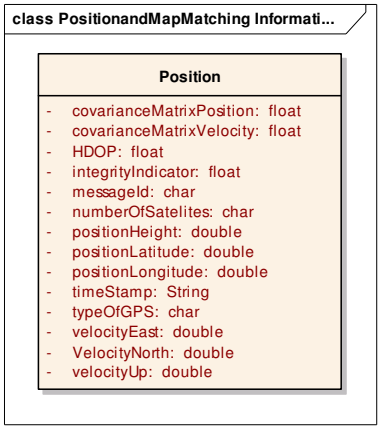

Figure 58: Position information

4.1.4 Interaction model

An example interaction of the position and map matching facility is shown in Figure 59.

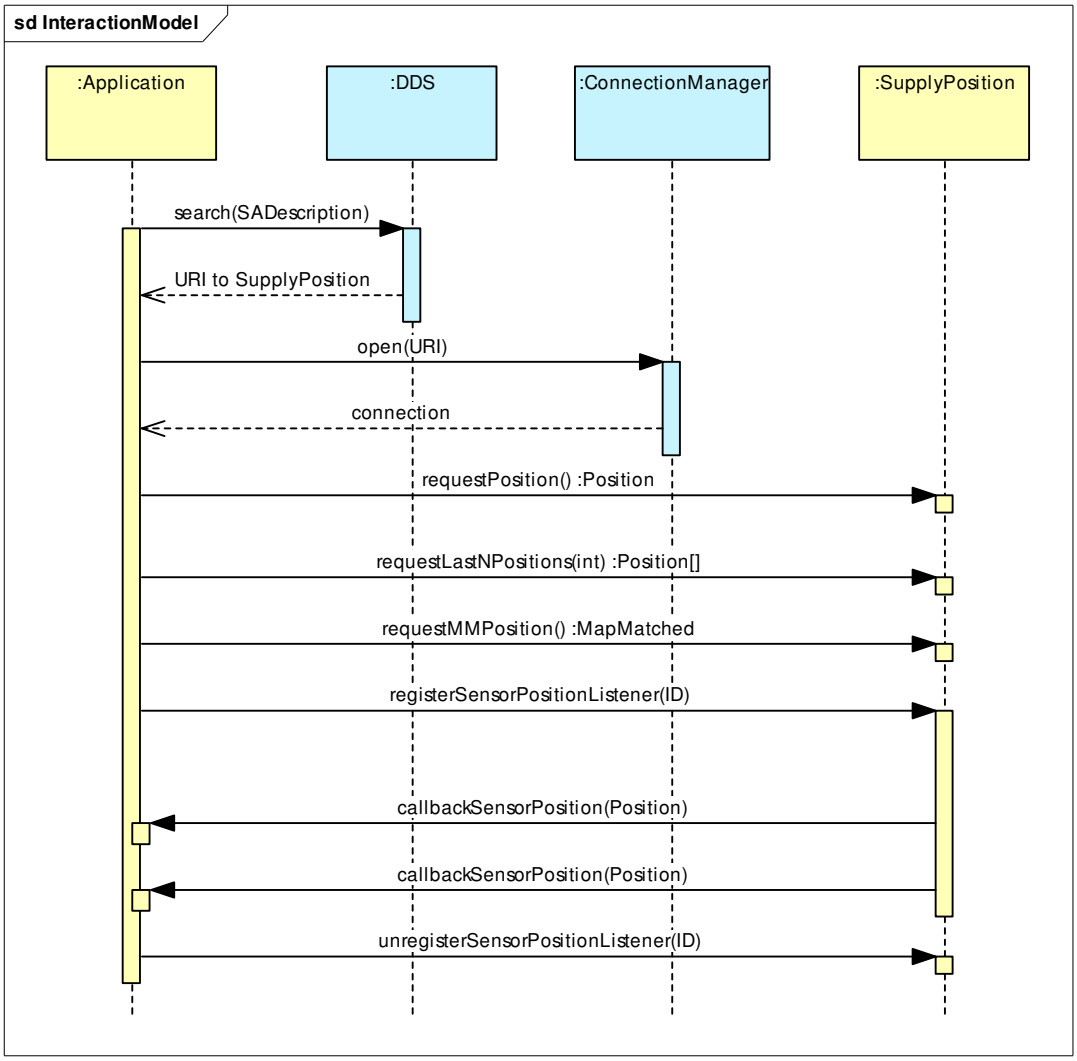

Figure 59: Position and map matching example scenario

An application first needs to look up the supply position to use the provided services of the position and mapping interface. The distributed directory service facility (see section 3.2) is used for look up. Then the connection manager facility (see section 3.5) is used for setting up a connection. When the connection is in place the application may use the provided position and map matching services as shown in Figure 59.

4.1.5 High level composite architecture

The high level composite architecture for the CVIS position and map related facilities provided by the POMA sub-project is shown in Figure 60. This component structure includes the components and interfaces for the following domain facilities:

Position and map matching.

Infrastructure position.

Map provision.

Location reference.

Geo-spatial platform.

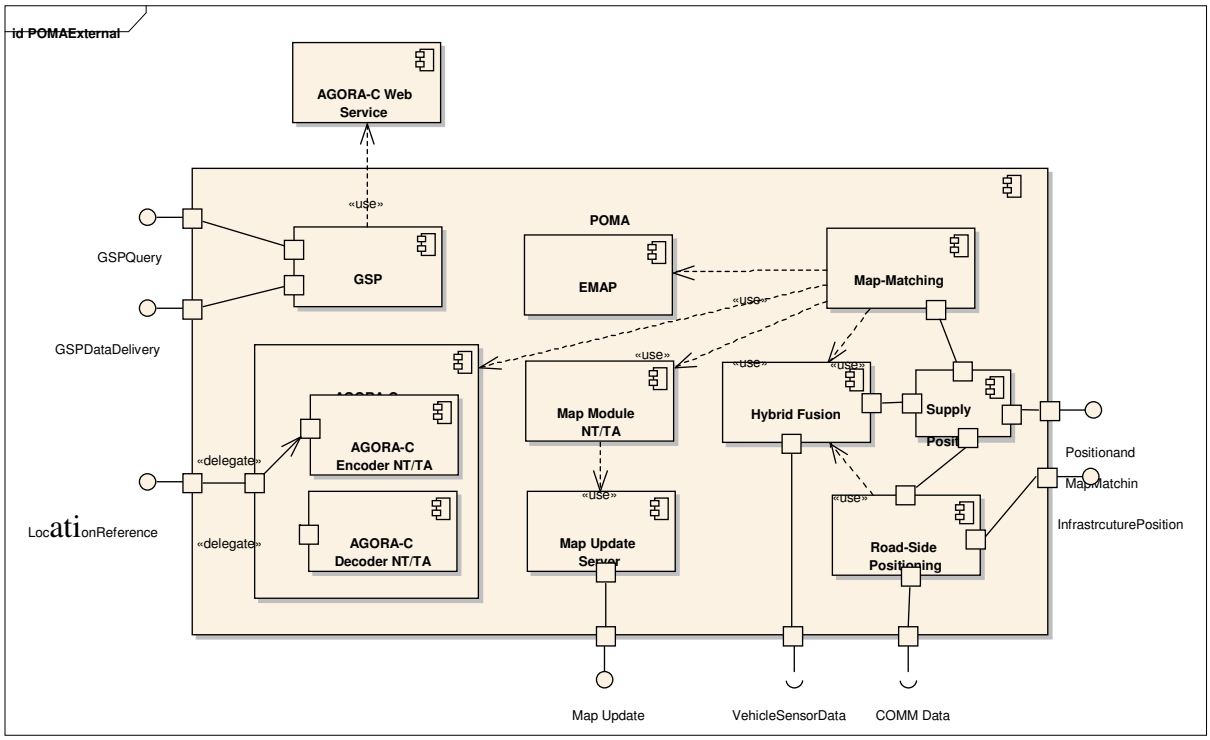

Figure 60: High level composite architecture for position and map related facilities.

4.2 Infrastructure position

This sub-section is based on the D.POMA.3.2 specification document. In this document the focus is on the external interfaces. For discussions of the internal architecture of the infrastructure position we refer to the D.POMA.3.2 specification document.

4.2.1 Overview

The infrastructure position facility provides the position of CVIS nodes by localizing them using the wireless sensor network. This involves also calibration of the system.

The CVIS applications typically do not access this facility directly but, rather it is designed to support requirements of the cooperative traffic information facility (see section 4.6) Positioning data provided by the infrastructure system will be available to the applications through the position and map matching facility as presented in the previous section.

CVIS nodes are located along the road, and with the availability of sensor units that can potentially be distributed over a wide area, localization of the single unit shall not be a manual configuration task.

Some anchor node with fixed position will provide absolute position, whereas the other nodes can measure some range information from distributed nodes. The range measurements (RSSI) will then be collected and used to define the position of all the nodes. Some anchor node can

also be mobile; this will allow having more accurate positioning and requiring less anchor nodes to be present.

The use case model of the Infrastructure position facility is shown in Figure 61.

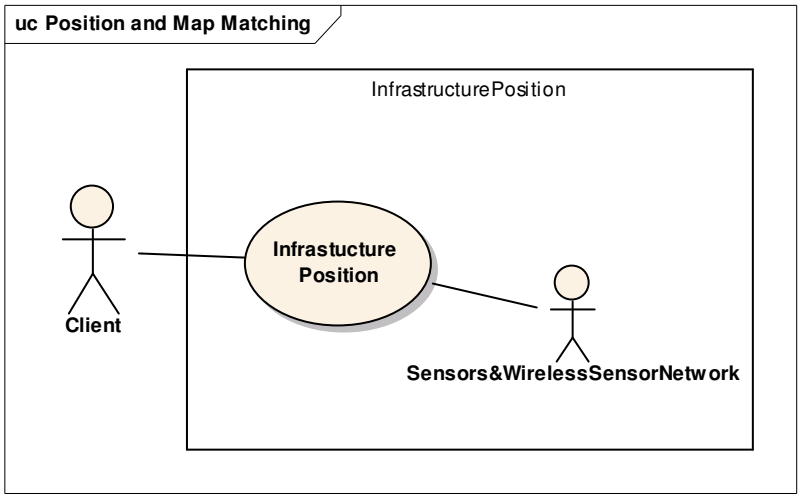

Figure 61: Infrastructure position use case model

4.2.2 Application programming interface

The API of the infrastructure position facility is shown in Figure 62.

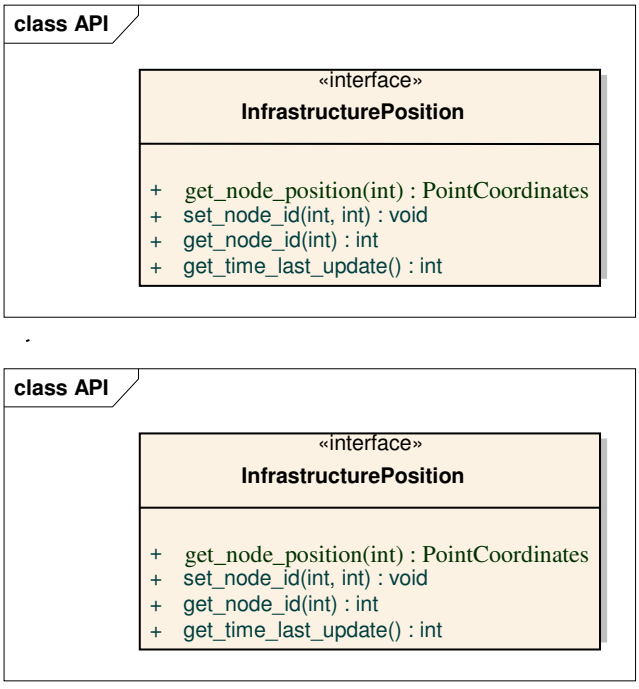

Figure 62: Class diagram for the infrastructure positioning of CVIS objects interface.

The following table describes the methods of the API.

Method	Parameters	Returns	Description
get_nod_position	Node_id: int	Latitude: double	The function returns

Method	Parameters	Returns	Description
		Longitude: double Accuracy_horizontal: double Time_last_update :date	the position, with associated information, of the requested node
set_node_id	Node_id:int Sensor_id:int	Result: boolean	Create a new node entry and Set the association between the node and the sensor
Get_node_id	Node_id	Sensor_id Node_id	Get the ID of the sensor associated to a node; get the node_id if the node exists
get_last_time_update	Node_id	Time: date	Return the last update event time of the node position

4.2.3 Information model

Figure 63specifies the information model of the infrastructure position facility.

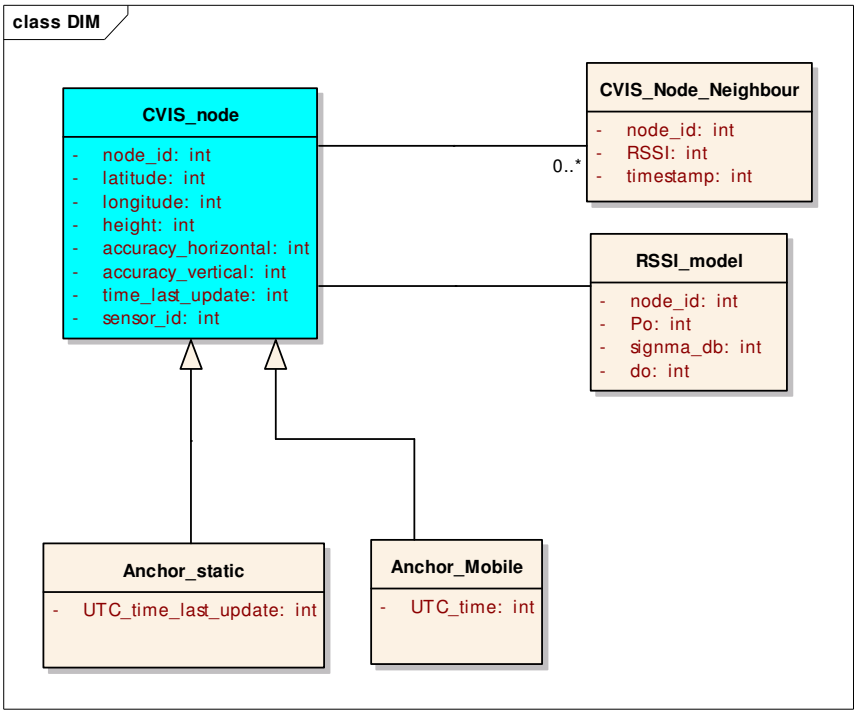

Figure 63: Information model for the infrastructure positioning of CVIS objects interface

4.2.4 Interaction model

An example interaction showing the usage of the infrastructure position facility is shown in Figure 64.

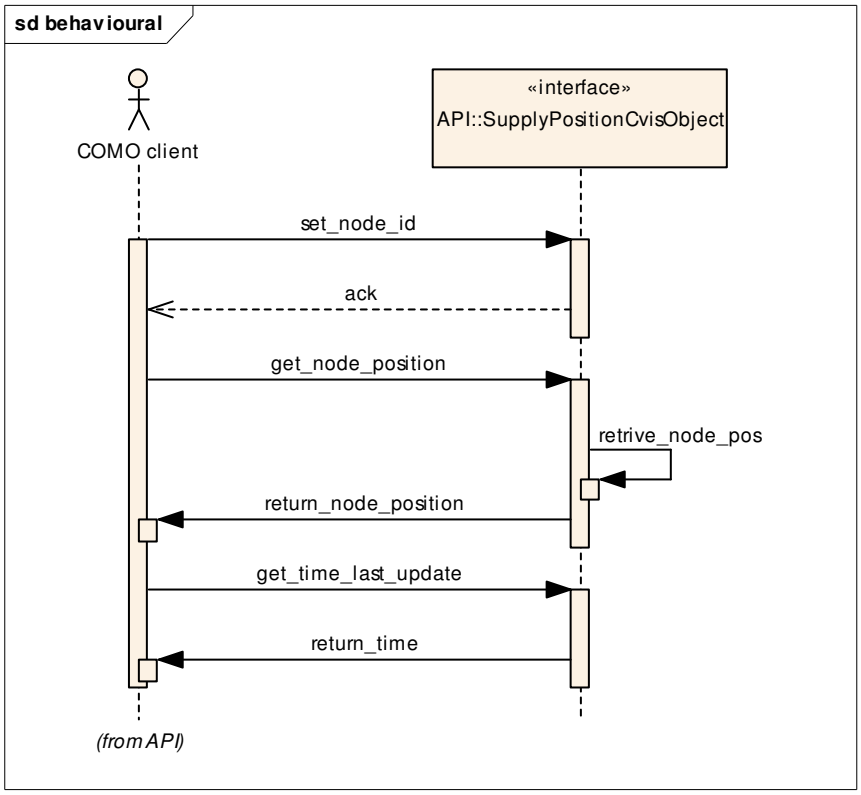

Figure 64: Interaction model for the infrastructure positioning of CVIS nodes

4.2.5 High level composite architecture

See Figure 60.

4.3 Map provision

This sub-section is based on the D.POMA.3.2 specification document. In this document the focus is on the external interfaces. For discussions of the internal architecture of the map provision facility we refer to the D.POMA.3.2 specification document.

4.3.1 Overview

The map provision facility provides services for supplying and updating maps as illustrated in the use case model of Figure 65.

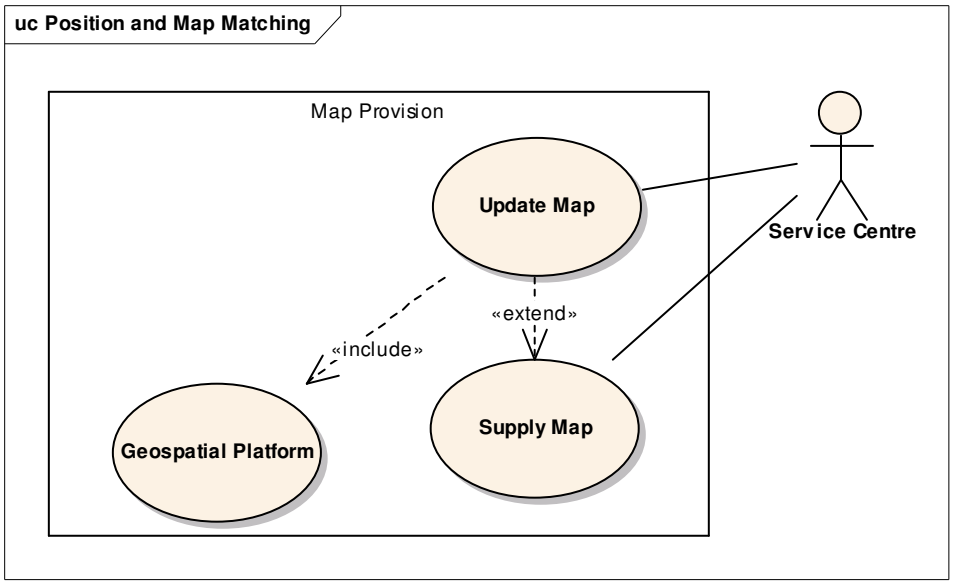

Figure 65: Map provision use case model

The SupplyMap use case will be supplied with the CVIS vendor specific information as agreed in WP2.

The UpdateMap will provide map updates according to the ActMap update exchange format. It represents a standardised intermediate XML format based on the data model of the ISO standard "Geographic Data Files" (GDF) designed for exchanging map updates between the proprietary formats of the map update suppliers and the map update users.

Constraints

The integration of the ActMap updates into the binary database (PSF) used by the CVIS facilities or applications requires an ActMap client capable of compiling the updates in an embedded environment. It is under the responsibility of the service or application to develop such an ActMap client to make use of the updates provided by the map update server. Another possibility is to directly provide PSF tiles for download. This has the advantage that the updates are compiled on the server side where a more performance compliant environment is available (see Figure 66). This alternative map update method can be provided by map engine providers to services or applications that require updates but have no ActMap client at their disposal.

In the following only the standardized ActMap process will be described. Other update processes offered by map providers, like PSF download, use proprietary interfaces. Map providers are in charge of providing these proprietary interfaces documentation to the CVIS application developers.

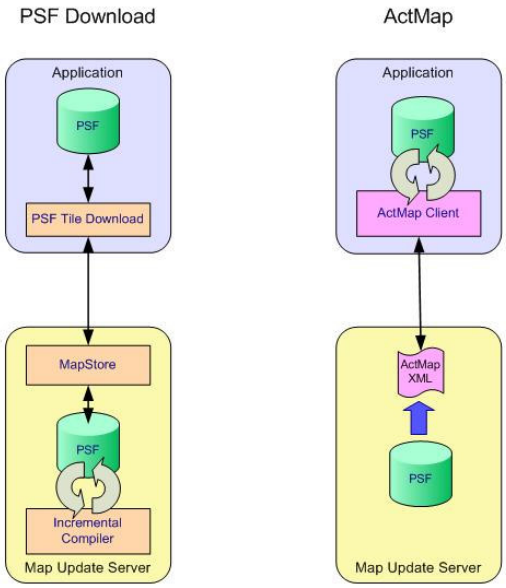

Figure 66: Distinction between the PSF download & ActMap update mechanisms.

4.3.2 Application programming interface

The general structure of the XML schema for the ActMAP update exchange format is provided in the ActMap specification chapter 2.8 (http://www.ertico.com/en/sub-projects/actmap/public_documents/). The detailed source code of the schema can be found in appendix A of the ActMap specification.

The ActMap map updates are stored on the ActMap server where the map updates can be accessed using an XML query.

A set of transactions can be given to the ActMap server as a standard XML request in order to optimize the updating process for the component that updates the infrastructure or in-vehicle map:

Provide the information for the baseline map and update supplier when the latest update was put on the server;

Give all updates that fulfil specific selection and filter criteria;

Give the size of the updates that fulfil specific selection and filter criteria.

The ActMap server considers three different types of requests:

Update discovery to determine whether the baseline map that is used to create the in-vehicle PSF is supported;

Update provision requests for getting map updates and corresponding information;

Profile requests to set general configurations for getting map updates.

Dependent on the nature of this XML request an XML file with a specified structure is provided back. The corresponding type of request - response combination can be found in the ActMap specification document.

The logical API of the map update interface is depicted in Figure 67.

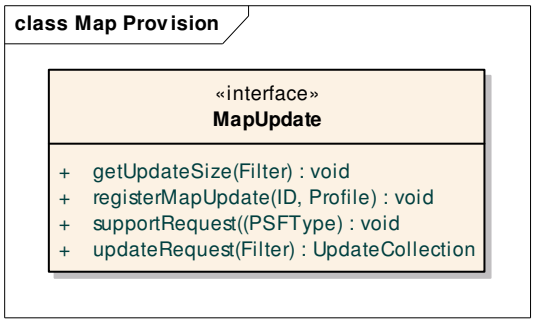

Figure 67: Map update API

4.3.3 Information model

The Figure 68 shows the data model for the map update facility. It contains Meta data for the corresponding baseline map and has several update collections.

Further information about the model can be found in the ActMap specification chapter 8.1.2, figure 15 which is at http://www.ertico.com/en/sub-projects/actmap/public_documents/.

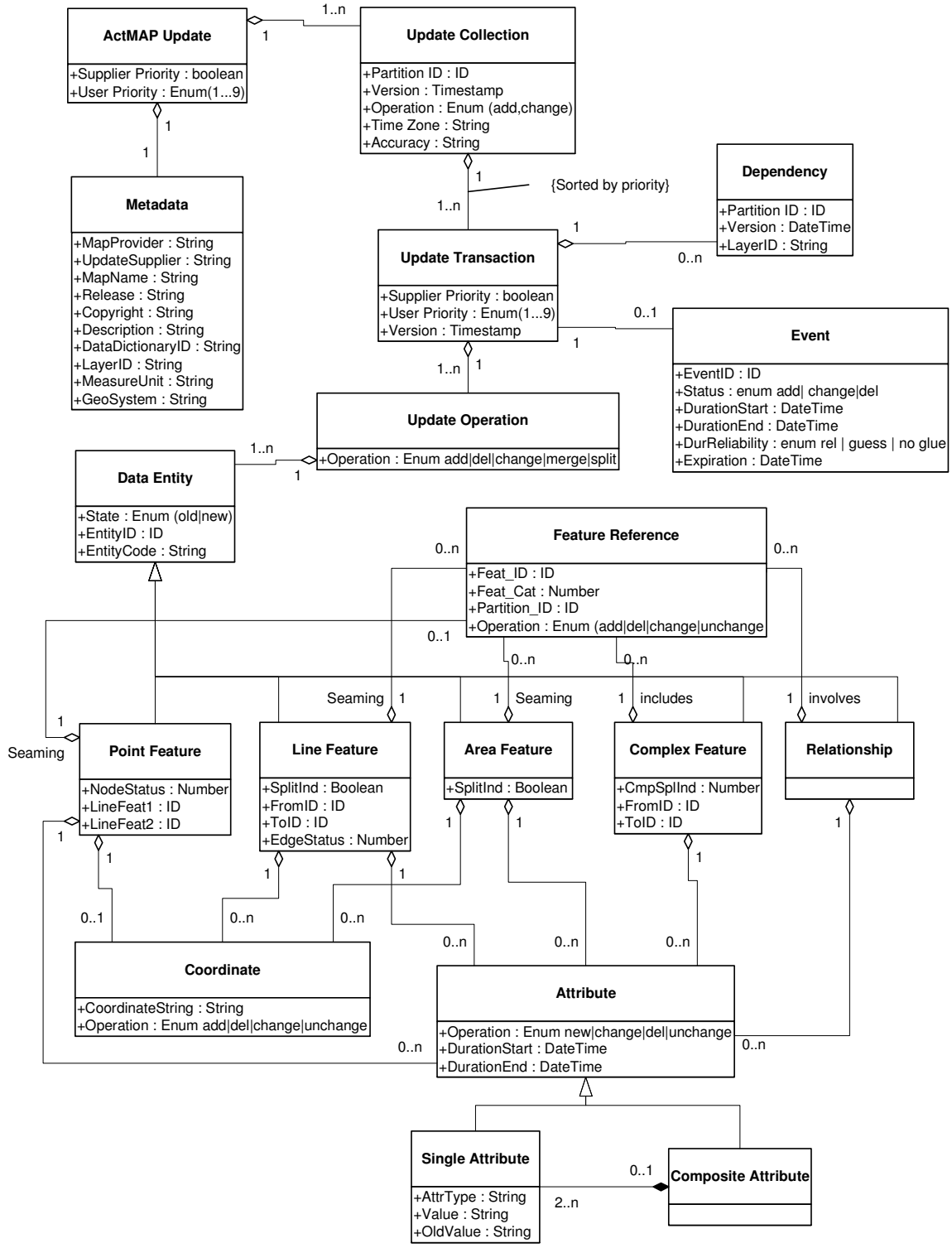

Figure 68: Information model for map update.

4.3.4 Interaction model

The following sequence diagram illustrates how the communication between the in-vehicle map update module including functions of the update trigger and update manager and the ActMAP system can look like. At the in-vehicle side there are three entities:

Update trigger: this is signalling a situation when an update is needed - the update trigger is a component needs to be implemented by the CVIS geo-engine module provider

Update manager: this is deciding which types of updates are necessary and is contacting the ActMAP service centre to get the updates - the update manager needs to be implemented by the CVIS geo-engine module provider.

Map: this is the onboard map database to be updated. The update manager will perform the updating of the map.

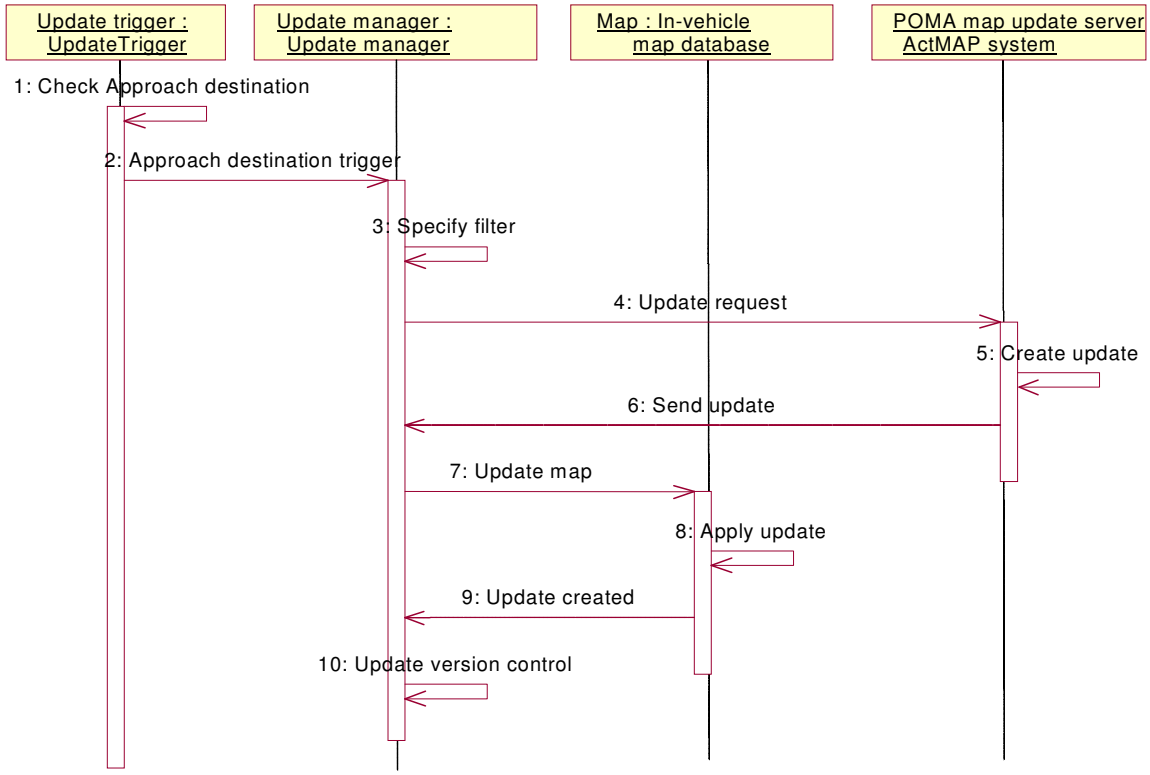

Figure 69: Behavioural model with process flows and timing

The process flow is done according to the map update needs of the in-vehicle platform. Nevertheless the same approach can be taken into account for the infrastructure.

In the process two different mechanisms - PUSH and PULL - can be taken into account. Both mechanisms are possible. Nevertheless in CVIS at least one mechanism will be enabled according to application needs. In general in case the PUSH mechanism is implemented a bidirectional communication is needed at least for subscription.

Further information about the subscription model can be found in the ActMap specification

chapter 6.2, figure 15, (http://www.ertico.com/en/sub-projects/actmap/public_documents/).

4.3.5 High level composite architecture

See Figure 60.

4.4 Location reference

This sub-section is based on the D.POMA.3.2 specification document. In this document the focus is on the external interfaces. For discussions of the internal architecture of the location reference facility we refer to the D.POMA.3.2 specification document.

4.4.1 Overview

This facility allows a CVIS facility or application to encode or decode an AGORA-C location reference.

Map matched positions and geo-referenced data will be exchanged between independent CVIS components as AGORA-C Strings. The location reference facility provides a service to encode and decode location referenced data. Locations to be encoded consist among other information of a list of map link identifiers and are therefore map provider dependent. From an implementation point of view, the location reference facility will provide 2 AGORA-C encoders and decoders, one for each map provider. Depending on the map used by the CVIS facility or the application, the corresponding encoder/decoder must be chosen. Both AGORA-C services will use the same interface and therefore calls and parameters will be identical.

The following description is provisional. To encode map-matched positions POMA will use a simplified AGORA-C which will be defined in the SAFESPOT project. The interface will be adapted to this AGORA-C in alignment with SAFESPOT. Also, lane position is not considered here and will have to be added.

The use case diagrams in Figure 70 and Figure 71 shows main use cases of the location reference facility. The AGORA-C encoder and decoder actors perform the calculations using either the location data or link identifier inputs provided by the clients.

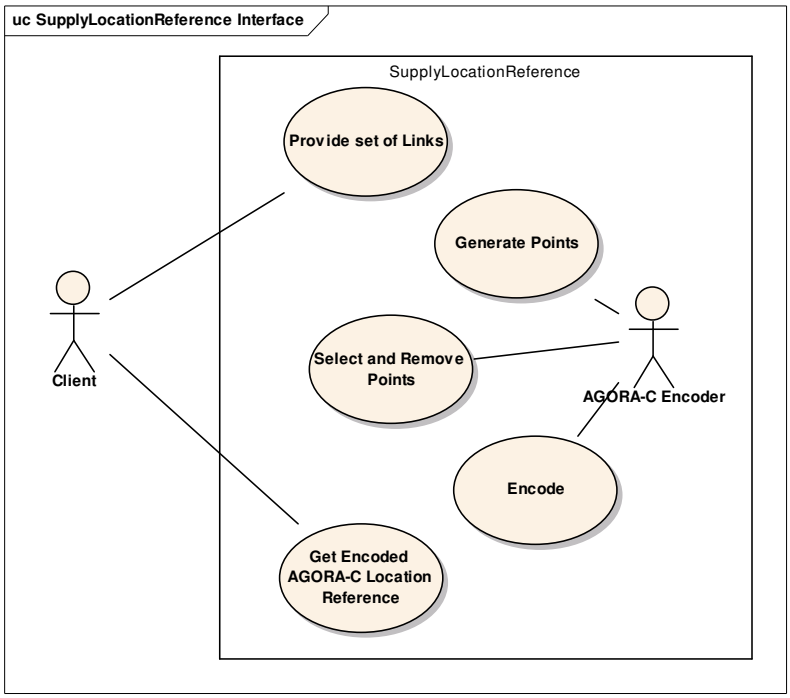

Figure 70: Use case model for the SupplyLocationReference Interface. This UC describes the AGORA-C encoding interactions.

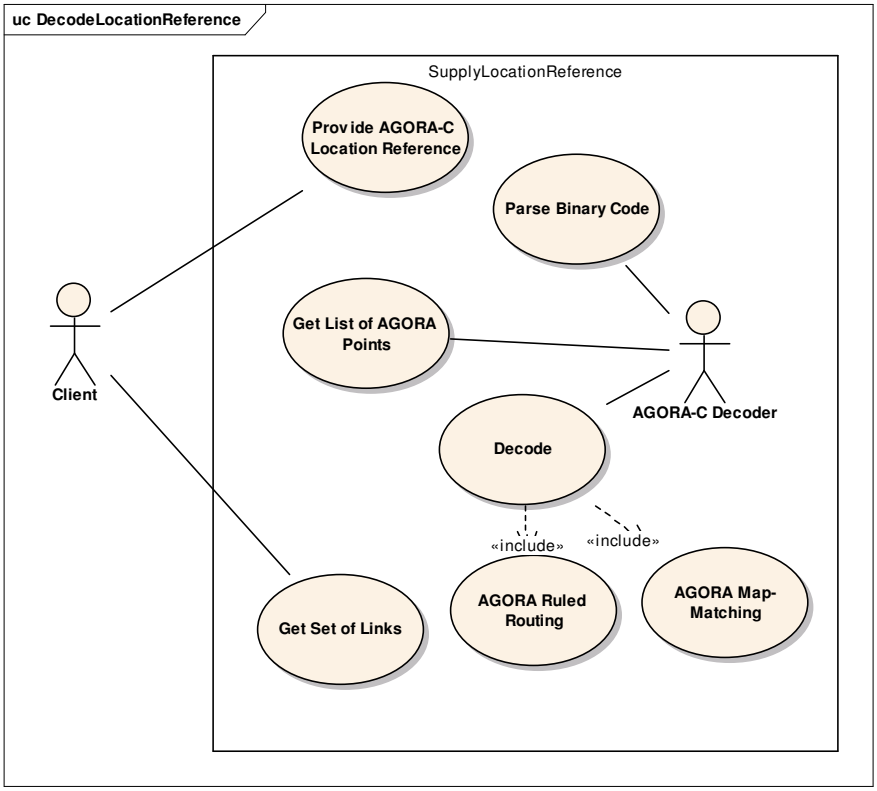

Figure 71: Use case model for the SupplyLocationReference interface. This UC describes the AGORA-C decoding interactions

4.4.2 Application programming interface

The API of the location reference facility is shown in Figure 72.

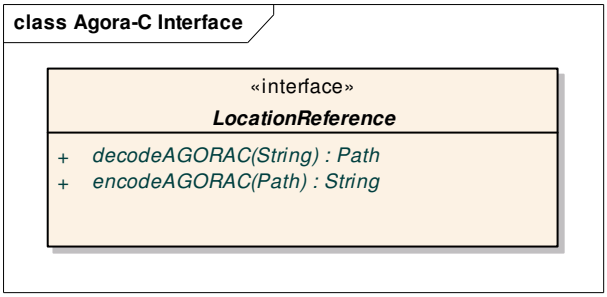

Figure 72: The LocationReference API.

The following methods are available for the location reference facility.

Method	Parameters	Returns	Description
encodeAGORAC()	listOfLinks : List<Link> startOffset : int endOffset : int	String	Constructs an AGORA-C String from a list of directed Links, a startOffset and an endOffset.
decodeAGORAC()	agoracCode : String	listOfLinks : List<Link> startOffset : int endOffset : int	Takes an AGORA-C String as input and retrieves the list of links with link IDs and direction that constitute the encoded path.

For both methods, the startOffset is the distance from the starting point of the first link of the list to the point where the actual location to be encoded starts.

Similarly, the end offset is the distance from the end of the path to the end of the last link.

4.4.3 Information model

The information model of the location reference facility is shown in Figure 73

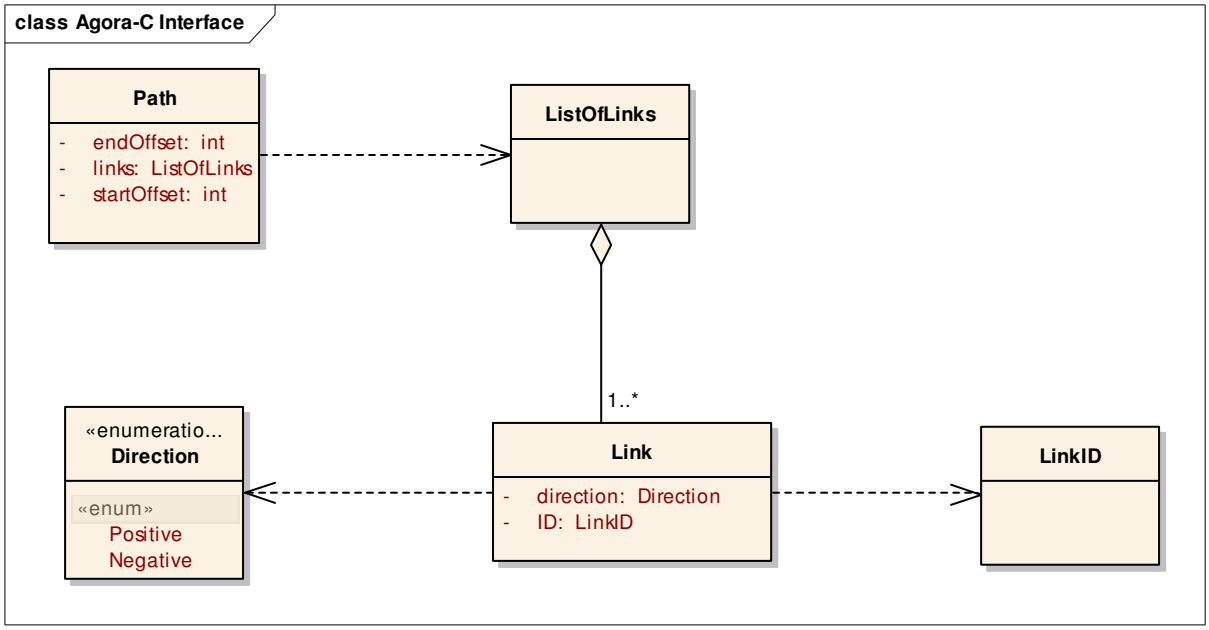

Figure 73: Location reference information model

4.4.4 Interaction model

The interaction model of the location reference facility is shown in Figure 74

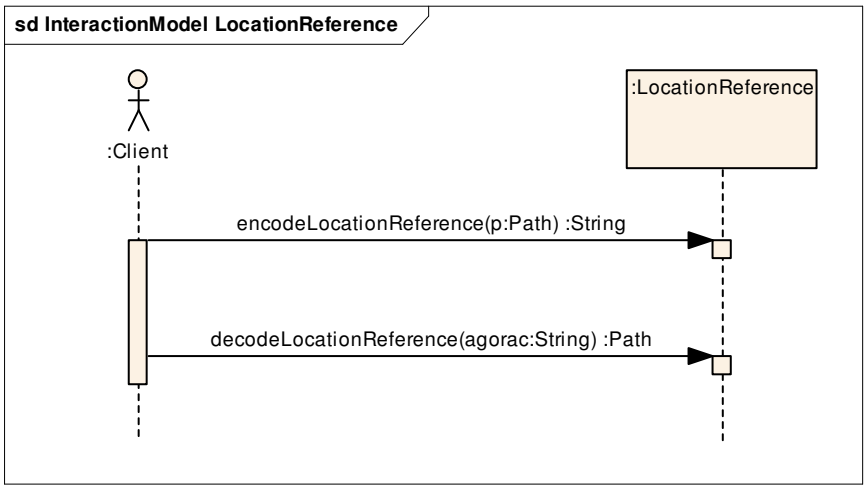

Figure 74: Location reference interaction model

4.4.5 High level composite architecture

See Figure 60.

4.5 Geo-spatial platform

This sub-section is based on the D.POMA.3.2 specification document. In this document the focus is on the external interfaces. For discussions of the internal architecture of the GSP facility we refer to the D.POMA.3.2 specification document.

4.5.1 Overview

The GSP facility has two external interfaces. The first one is used the query interface for accessing different GSP functionalities. The second one is used to deliver traffic data to the GSP.

The query interface provides GIS functions such as "Geocode", i.e. to convert an address to X,Y coordinates, "MapDisplay & Route", i.e. to calculate the optimal route from a single location to one or several destinations.

To enhance the information provided by the GSP, traffic data may also be added. This may then be used to enhance the response provided by "MapDisplay & Route" requests. Traffic data is added through what is denoted the GSP data delivery interface.

The geo-spatial platform provides location information, geographic information and functionality as an asynchronous messaging-based Internet product.

The GSP is exposed as a service using an XML-based framework which allows the user to interact with it by sending requests and receiving responses using HTTP/1.1.

The use case diagram in Figure 75 specifies the main services provided by the GSP facility.

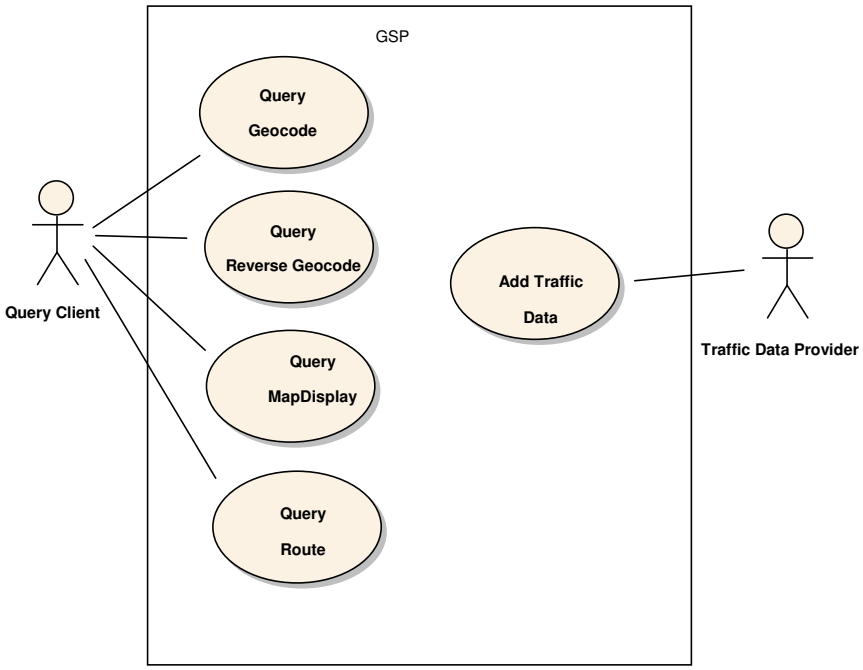

Figure 75: Use case model for the GSP query interface.

4.5.2 Application programming interface

The two interfaces provided by the GSP facility is specified in Figure 76. The GSP query API provides geo-coding, map display and routing information, the GSP data delivery API enables traffic data to be fed in from external sources. This traffic information will then be utilized by the GSP facility to provide more enhanced information through the GSP query API.

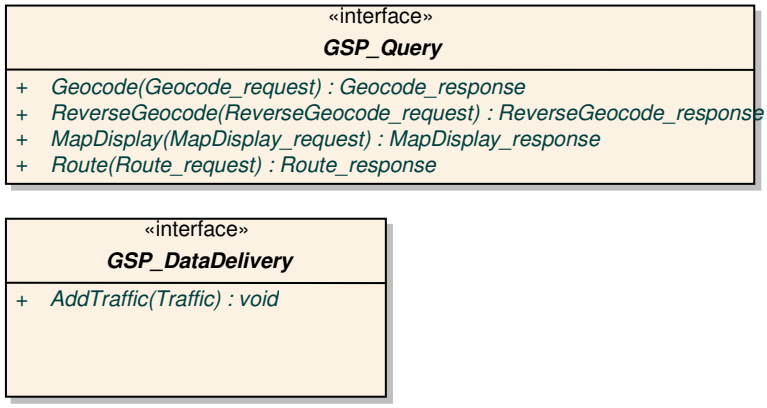

Figure 76: Class diagram for the GSP query interface.

The methods provided by the GSP query API are elaborated further in the table below.

Method	Parameters	Returns	Description
Geocode	Address: string NumResultRows: int	locationRef: AGORA-C score: int address: string	Returns a location reference for an address
ReverseGeocode	locationRef: AGORA-C	address : string	Returns an address for a location reference
MapDisplay	locationRef: AGORA-C mapStyle: MapStyle traffic	urlRef: string filename: string	Returns a link (URL) to a map image
Route	start: AGORA-C end: AGORA-C	time :double distance: int stages: ListOfStages	Returns a route between 2 points using static map data

The GSP query API requires all location references to be in AGORA-C, where the AGORA-C web service will be used for encoding and decoding.

The request and response are in XML format. The full specification is available in the GSP location agent specification.

Methods and parameters to access the GSP data delivery API are presented in the following table.

Method	Parameters	Returns	Description
AddTraffic()	Traffic	void	Add traffic data from COMO to the GSP

4.5.3 Information model

Figure 77 specifies the information model the GSP facility.

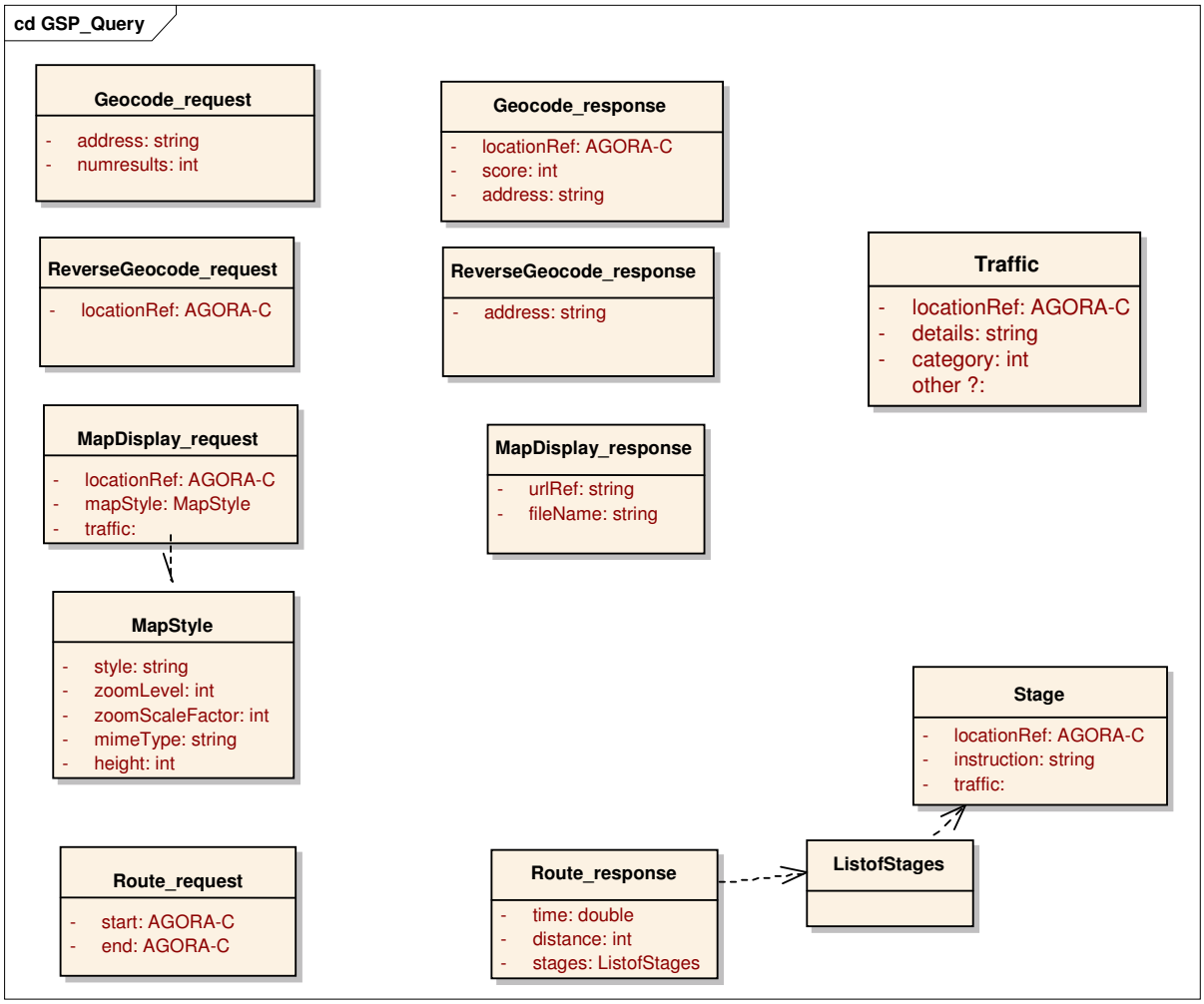

Figure 77: Information model for the GSP query interface.

4.5.4 Interaction model

A typical GSP query transaction may be summarized by some or all of the following steps:

An application constructs a standard XML request based on the interface defined by the GSP query API.

Depending on the nature of this XML request, GSP may do one or more of the following:

Geo-code an address;

Reverse geo-code a location, i.e. to convert X,Y coordinates to corresponding address;

Request a custom rendered map image;

Derive a route between two points.

GSP then responds to the application with the appropriate results in the form of an XML response.

In cases such as map request where large amounts of geographic data are produced, the raw data is output to a dispatch demilitarized zone and the reference location of that information is included in the XML response for the application to download.

Example scenarios related to the four methods of the GSP query API are shown in Figure 78, Figure 79, Figure 80 and Figure 81.

Geocode method

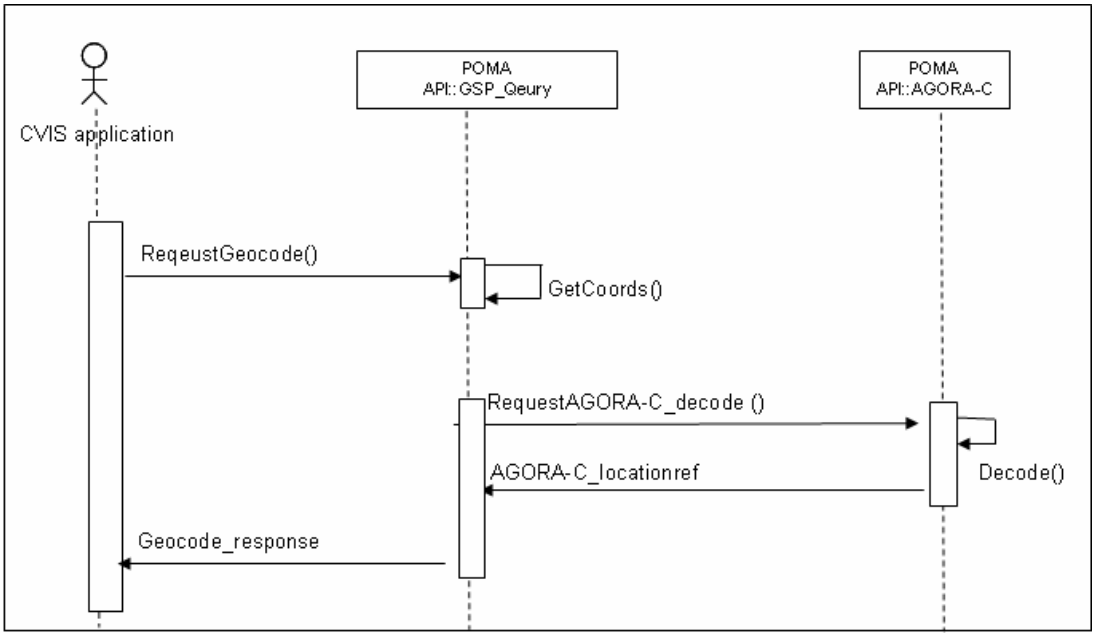

Figure 78: Behavioural model for the geo-code method of the GSP query interface.

Reverse geo-code method

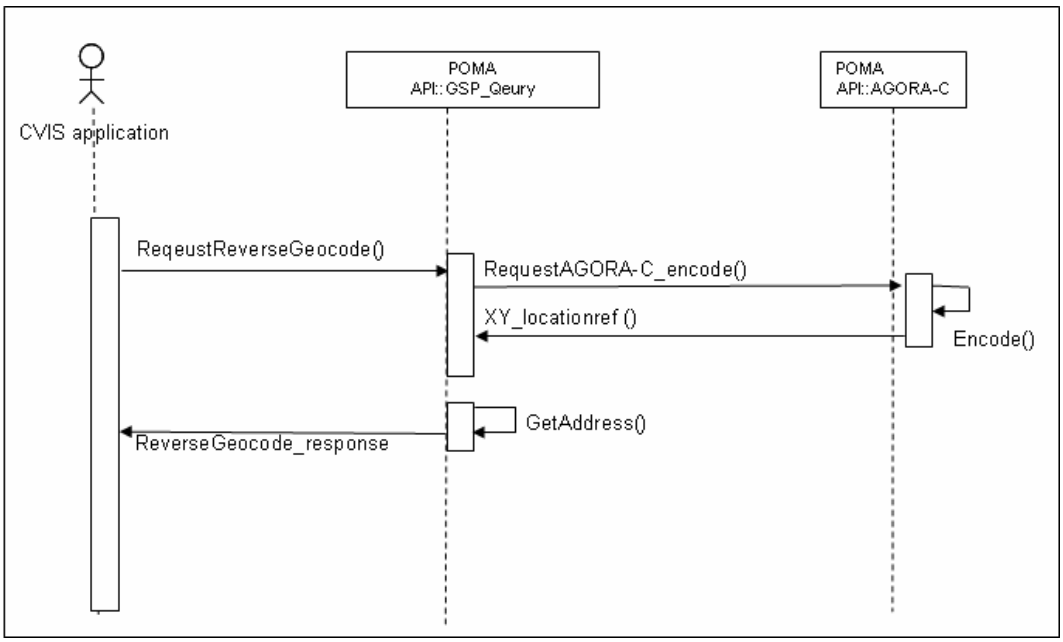

Figure 79: Behavioural model for the reverse geo-code method of the GSP query interface.

Map display method

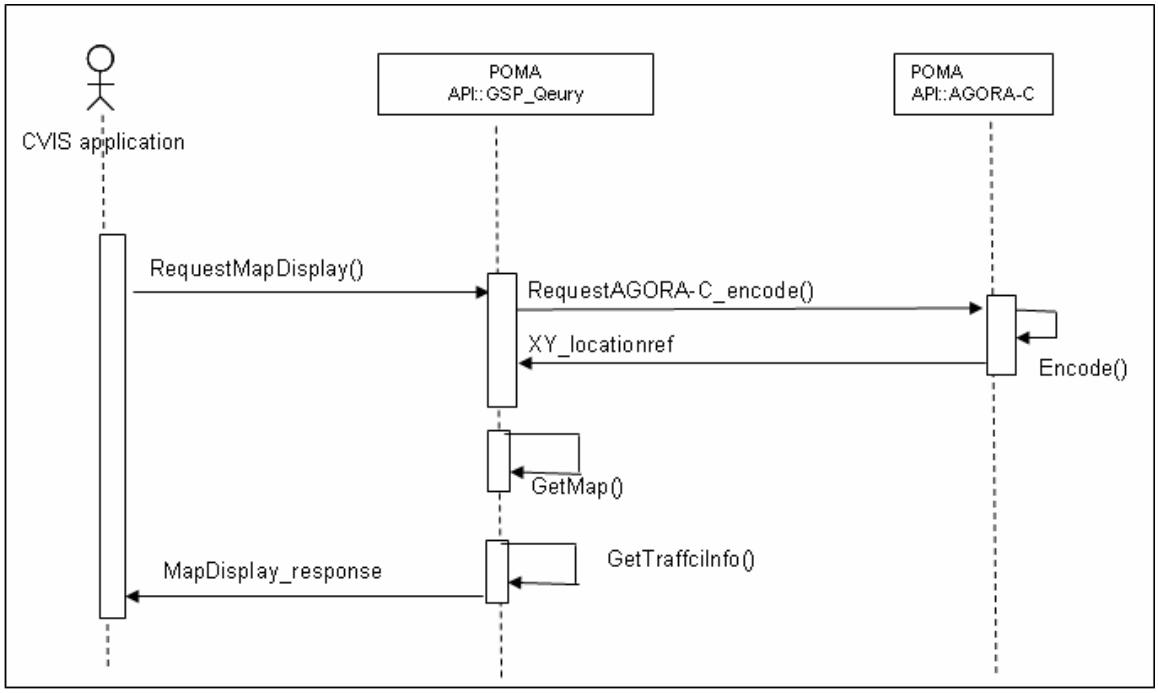

Figure 80: Behavioural model for the Map display method of the GSP query interface.

Route method

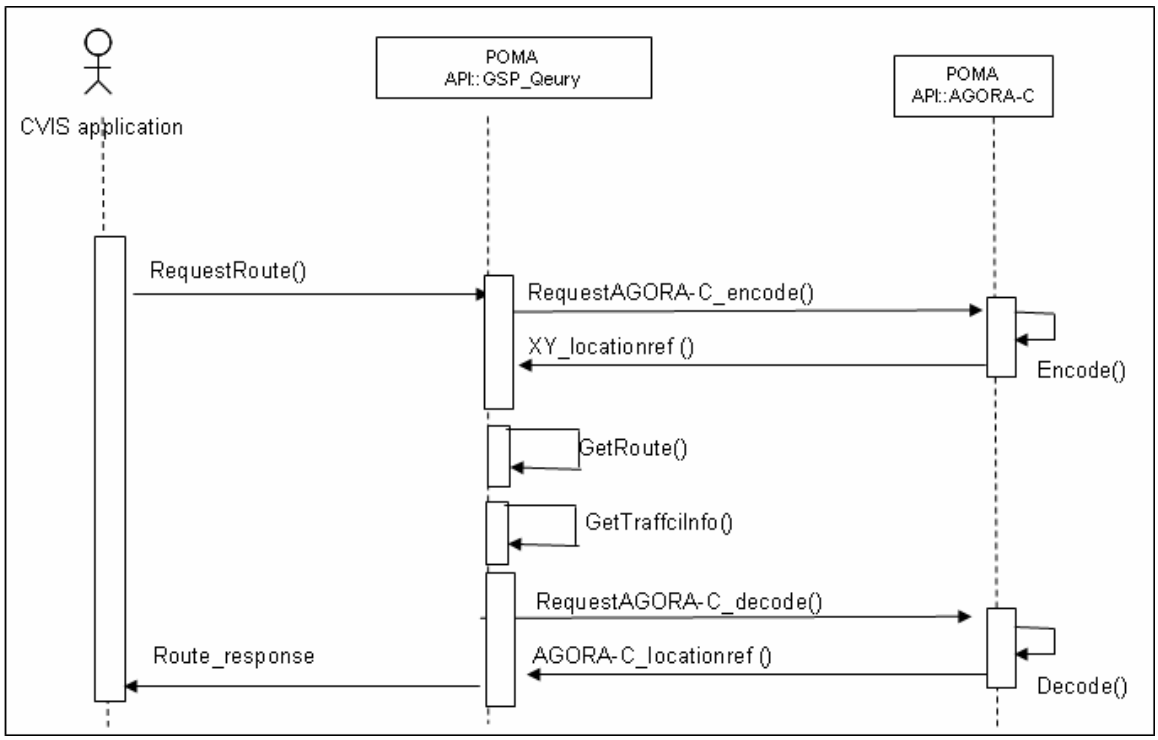

Figure 81: Behavioural model for the route method of the GSP query interface.

The GSP data delivery interaction model is shown in Figure 82.

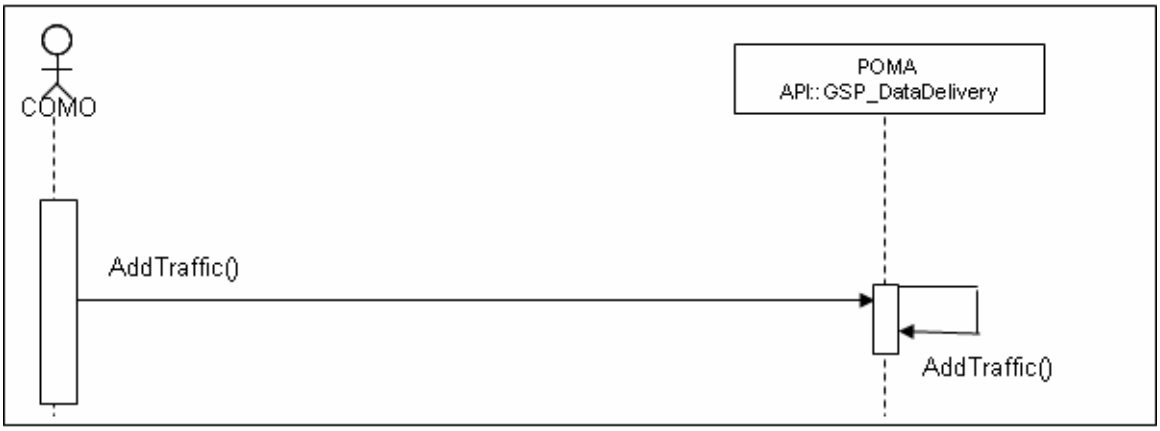

Figure 82: Behavioural model for GSP data delivery interface.

4.5.5 High level composite architecture

See Figure 60.

4.6 Cooperative traffic information

"COoperative MOnitoring" (COMO) has the task to define applications within CVIS which deliver an agreed set of traffic data to other CVIS applications. Since COMO is delivering a set of data and information which are to be used by numerous applications these data sets need to be commonly available within CVIS using agreed data formats and access mechanisms. The intention must be that every interested application knows where and how to access COMO data and how it looks like.

4.6.1 Overview

The following figure explains how COMO interacts with CVIS applications (produced by the applications sub-projects), with the geo-referencing functionality of POMA and with COMM. Of course, FOAM is the environment which provides the means to set up the interfaces by providing standard functionalities. Thus FOAM is not displayed here - just see it as the canvas on which this picture is drawn.

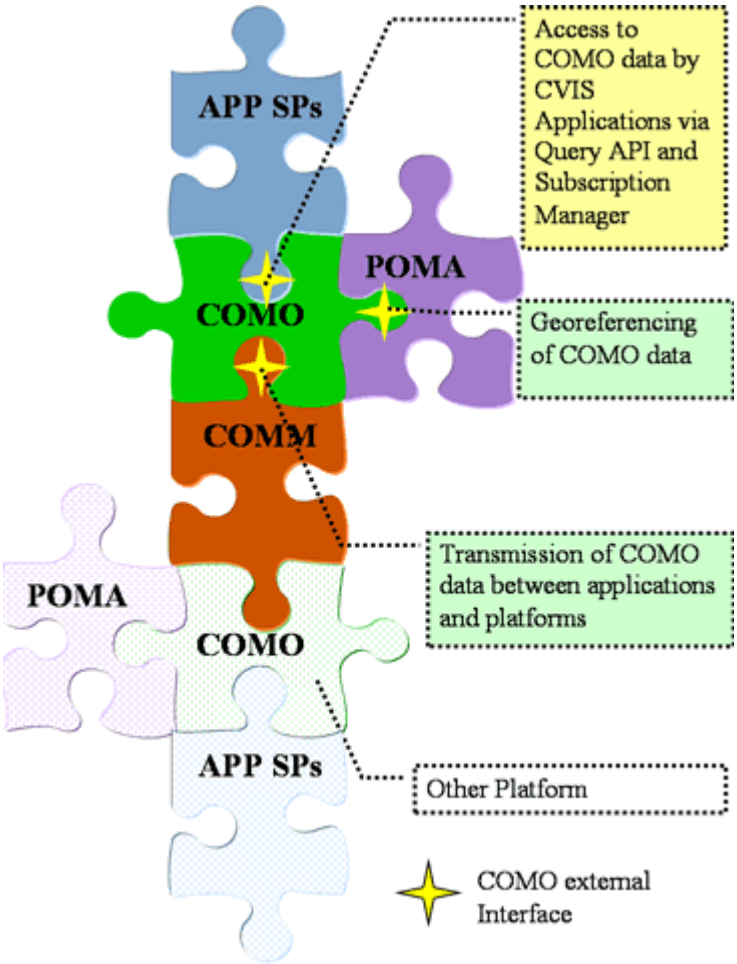

Figure 83: COMO system overview - COMO integration concept

As seen in Figure 83, COMO provides, apart from several interfaces between the COMO-internal applications, four external interfaces towards

- CVIS applications (designated as APP SPs) for retrieving COMO data via the COMO-

API, these are the applications from the CVIS Subprojects CINT, CURB, CF&F but also POMA and COMO, as well as for the LDM Q/T-API

- CVIS applications for the COMO subscription service (displayed together with the interface above)
- POMA for the geo referencing of COMO data and
- COMM for broadcasting and receiving COMO data from other platforms

4.6.2 Application programming interface

High level description of the interface:

The COMO API is the data access API of all COMO data including the LDM. On the one hand it provides together with the LDM Manager elementary data object related read, write, delete and insert methods to manage the data of the LDM. These access methods are named Q-API (read) and T-API (write, delete, insert). The only CVIS component that is allowed to use the T-API is the "COMO Data Fusion" and the "COMO Local Traffic State computation" (chapter 10).

Another functionality that is additionally provided by the LDM manager is the "Geo-spatial Query Engine". This component allows for geo-spatial querying of any data of the data base.

The following Figure 84 describes the related process. Any CVIS application can access data of the LDM data base by means of the COMO API. It simply has to register and submit the data request and receives in return the specified data from the API.

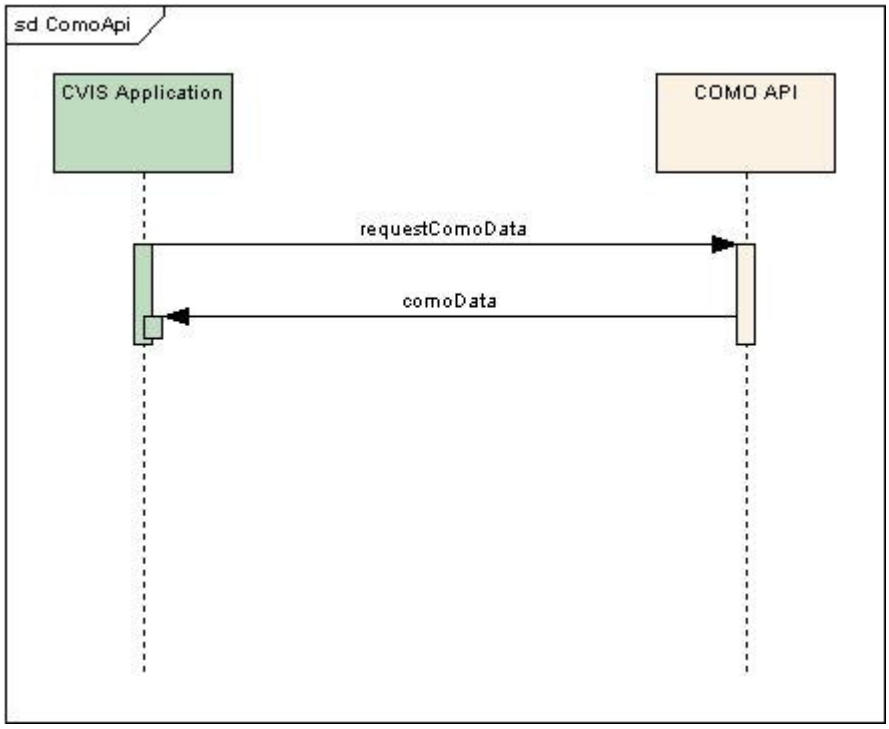

Figure 84: COMO API

The specific details of the information model were defined together with SAFESPOT. The access of data from the LDM will be realised through the COMO API via the SAFESPOT query API.

4.6.3 Information model

This section provides the specification of the COMO domain information model, which is the data structure of the LDM database. The domain information model identifies and defines the main concepts of the domain. This model typically embraces actors and information objects that flow within the system or application. The concept relationships are also important aspects and should be described. The concepts are modelled in terms of their types.

A detailed description of the data and information sets provided by COMO is contained in the annex under chapter 12.1 of D_COMO_3.2, Version 2.0.

Not all sets of information or data are necessarily available in all platforms. Vehicle sensor data, for example, might be pre-computed before they are sent to the CVIS platform, thus resulting in the fact that XFCD messages might be provided by the legacy system directly but vehicle sensor data will not. The availability of the COMO data is specific, depending on the legacy system manufacturer's policy constraints or the technological availability of the data in the legacy system. However, the COMO architecture takes care, that all COMO data can be contained in the COMO data sets if available from the legacy platform.

The COMO data set comprises traffic and environmental status data in the broadest sense, which also includes pre-processed sensor and detector data and the status of traffic control devices. As for the traffic status data, also incident information is contained. The environmental status contains besides weather also road-surface conditions.

As illustrated in Figure 85 below the whole COMO data model is derived from one geo-referenced object, which means that the main characteristic of all data classes is to have a spatial reference. Besides this the "COMO data world" is subdivided into two:

The "Static Objects", which describe the surrounding of the vehicle or road-side unit in terms of road-geometry (static digital map), road-furniture of any kind (detectors, actors, etc.) and "Reference Tracks", which are the possible ideal driving tracks at intersections.

The "Dynamic Objects", which comprises all geo-referenced information that is time dependent. Of course all moving objects, e.g. vehicles, and vulnerable road users, can be found here with their actual positions and speed. Furthermore, the traffic control, the traffic and the environmental states belongs to this object set.

Note that only the last status of dynamic objects is stored in the data base. Because of performance and limited memory reasons it is not foreseen to store historical data in the LDM data base.

In the case of the vehicle LDM additional requirements have to be considered that above all refer to the moving horizon of the vehicle. As a consequence all data instances have to be checked permanently whether they are outside the current horizon of the vehicle. All such data items must be deleted.

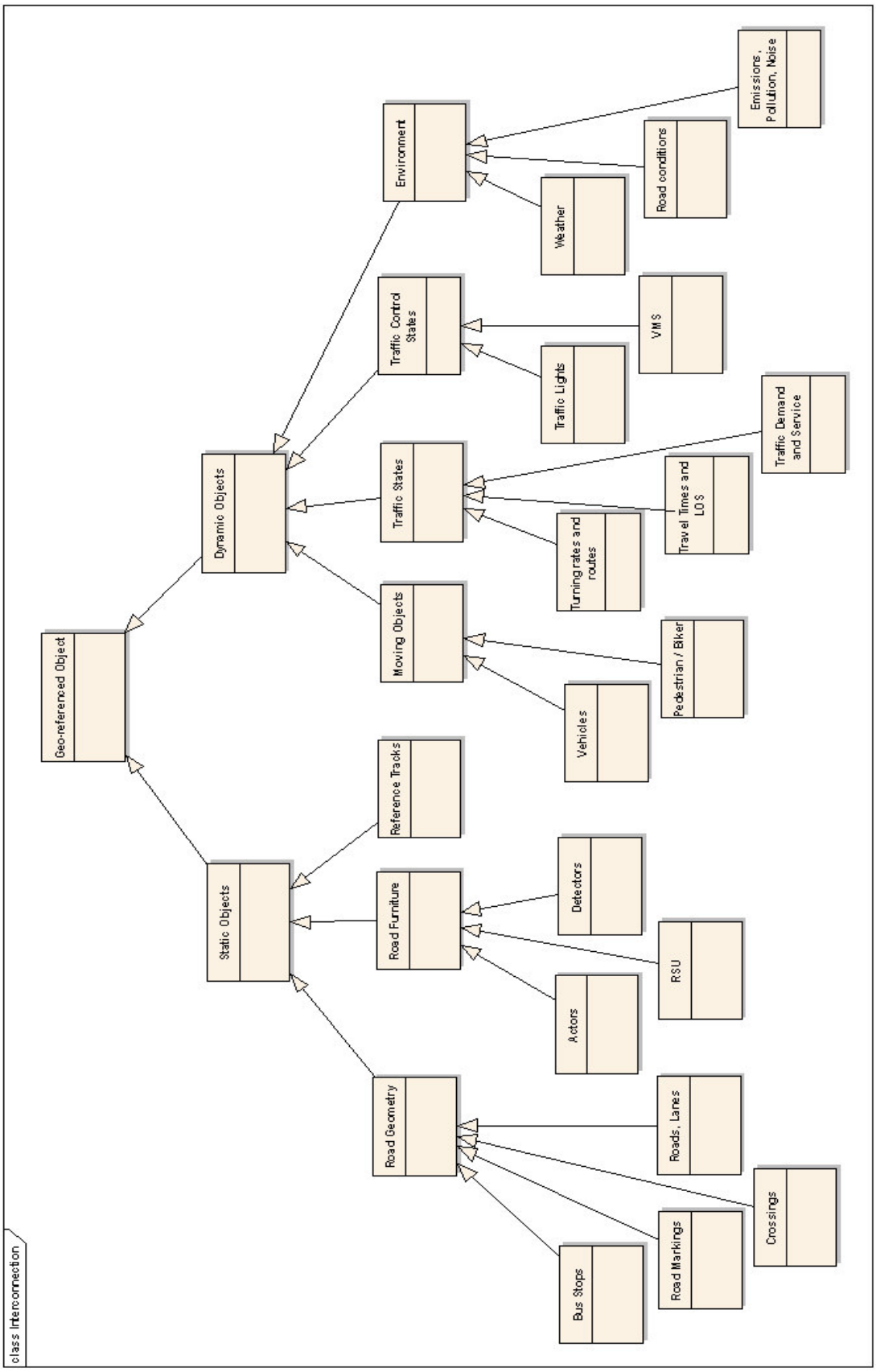

Figure 85: Overview of the COMO data model, showing the hierarchy of the data classes.

4.6.4 Interaction model

This chapter provides the sequence diagrams for the COMO data processing and fusion process and the LDM manager process.

COMO data processing / fusion

The figure below shows the general sequence for a data fusion in a COMO equipped CVIS platform.

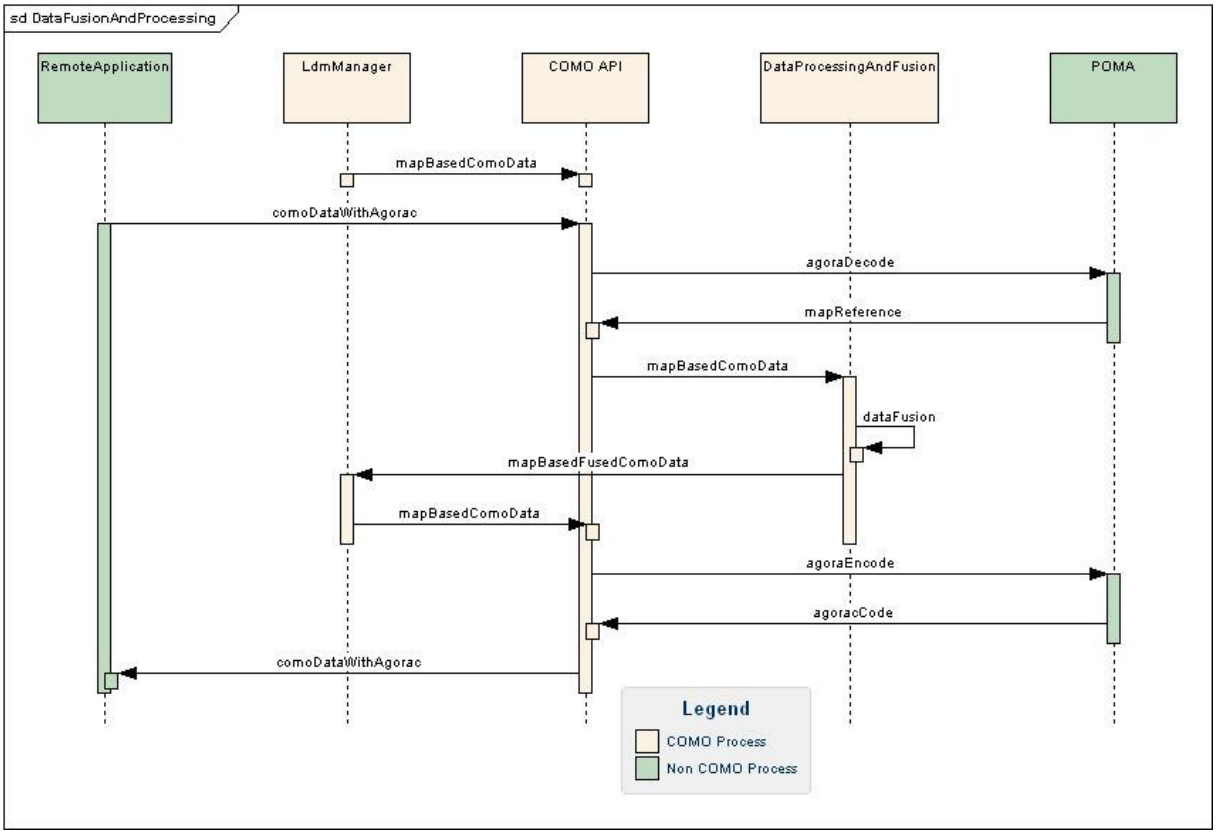

Figure 86: Data processing / fusion process

The COMO API receives input data from

- the LDM manager which provides access to the LDM holding also (unfused) high level (e.g. LOS) COMO data,
- remote applications providing relevant information via broadcast or unicast messages (e.g. XFCD)

Another data source is the legacy system the platform is connected to (not displayed in the picture above), which may provide information and data either to be entered into the LDM (as it would e.g. be the case with sensor data) or fused with data from other sources (as mentioned above).

The data fusion process is the core process of the data fusion package and responsible for the fusion of data from the sources mentioned above. The results of the data fusion process are written into the LDM using the LDM Manager and also provided through the COMO API.

Besides the traffic state calculation the data fusion package is the only CVIS process eligible

to write into the LDM. Thus also un-fused and sensor data are passing this process but are "only" forwarded to the LDM manager. While sensor data are utilised in the *computation of the local traffic state* process (one source for the data fusion package) and not in the data fusion package other (higher-level) data may not be fused if e.g. the LDM does not contain any other suitable data sets. As an example, there may only be one vehicle generating an EFCD message in the computation of the local traffic state, which in a vehicle can be the EFCD generation. If there is no other vehicle or RSU in the vicinity the LDM will not contain a suitable data set, e.g. another similar EFCD message.

COMO API

The COMO API is the data access API of all COMO data including the LDM. On the one hand it provides together with the LDM Manager elementary data object related read, write, delete and insert methods to manage the data of the LDM. These access methods are named Q-API (read) and T-API (write, delete, insert). The only CVIS component that is allowed to use the T-API is the "COMO Data Fusion" and the "COMO Local Traffic State computation".

Another functionality that is additionally provided by the LDM manager is the "Geo-spatial Query Engine". This component allows for geo-spatial querying of any data of the data base.

The following Figure 87 describes the related process. Any CVIS application can access data of the LDM data base by means of the COMO API. It simply has to register and submit the data request and receives in return the specified data from the API

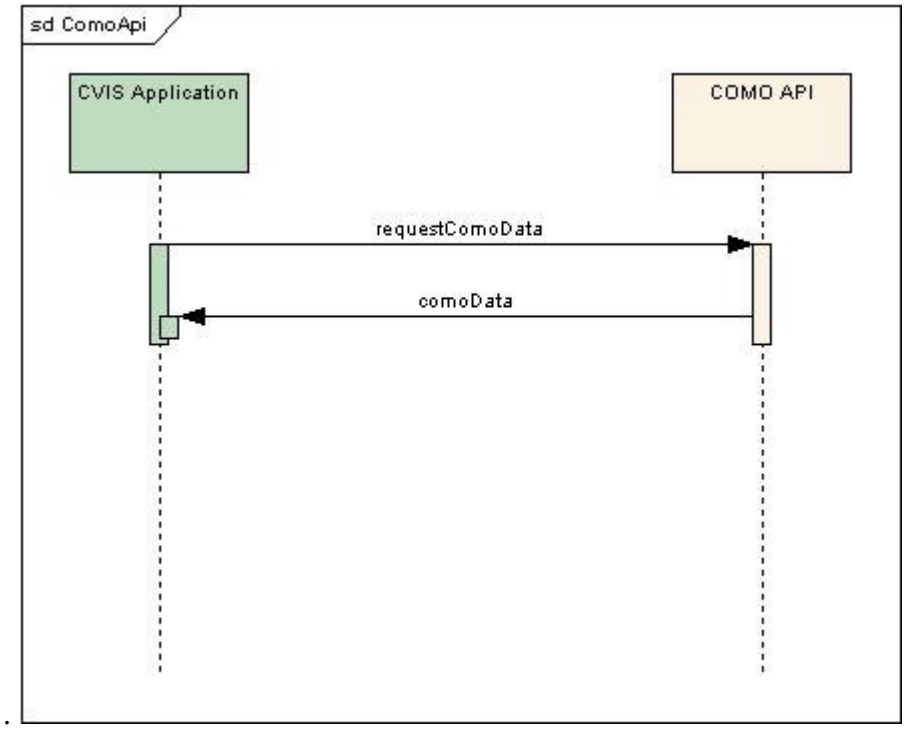

Figure 87: COMO API

The specific details of the information model were defined together with SAFESPOT. The access of data from the LDM will be realised through the COMO API via a SAFESPOT query API. The description of the LDM API is not part of this document.

The COMO API provides network transparent access for all CVIS SPs to information

provided by COMO. It realises functionalities which are responsible for the exchange of different kinds of information between applications on different hosts like vehicles, roadside units or service centres. COMO acts as a distributor to which data providers and data receivers can connect to. It realises a rule and constraint based Subscription and Notification service which manages the distribution of messages (COMO information) towards the different applications.

Data Providers, data receivers and the distributor

Figure 88 gives an overview how an application will fit into the COMO data distribution concept.

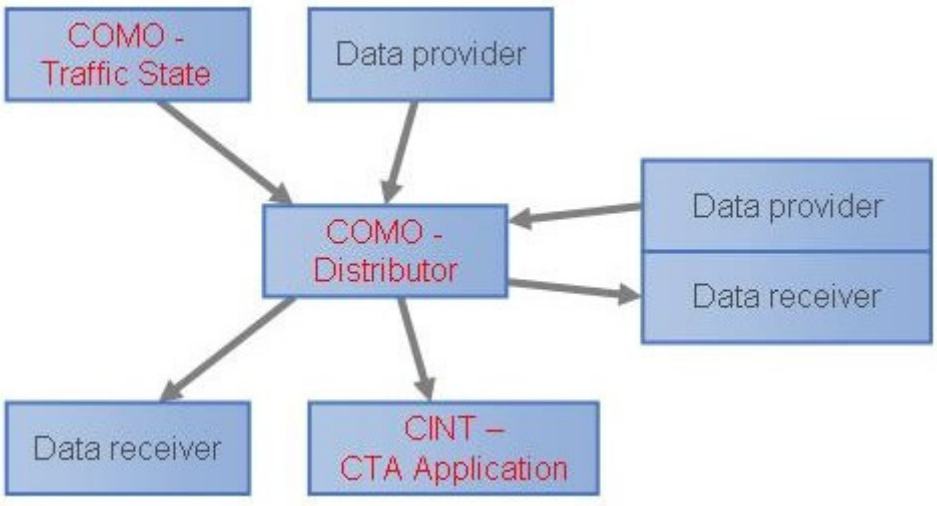

Figure 88: Applications & COMO API

This can be shortly described within the following use cases:

- Data receiver (CINT CTA Application) registers at the COMO Distributor (COMO API) in order to be informed when there is a new data unit of a certain kind (COMO Traffic State information)
- Data provider (COMO Traffic State) registers at the COMO Distributor as service for certain data units (COMO Traffic State information)
- Data receiver pulls a specific data unit
- Data provider pushes an updated data unit to all interested data receivers

COMO allows connecting data providers, data receivers or distributors over the network. This can be achieved by using the COMO client and server.

Network transparency

The COMO client provides an access to a COMO distributor over the network. More technically spoken, the COMO client and the COMO server act as stub and skeleton to a COMO distributor. Therefore a COMO client and a COMO distributor have the same API. This means that a data provider or a data receiver can interact with a remote distributor in the same way as if it was on the same machine. So it doesn't have to even need to know whether there is a network in between or not.

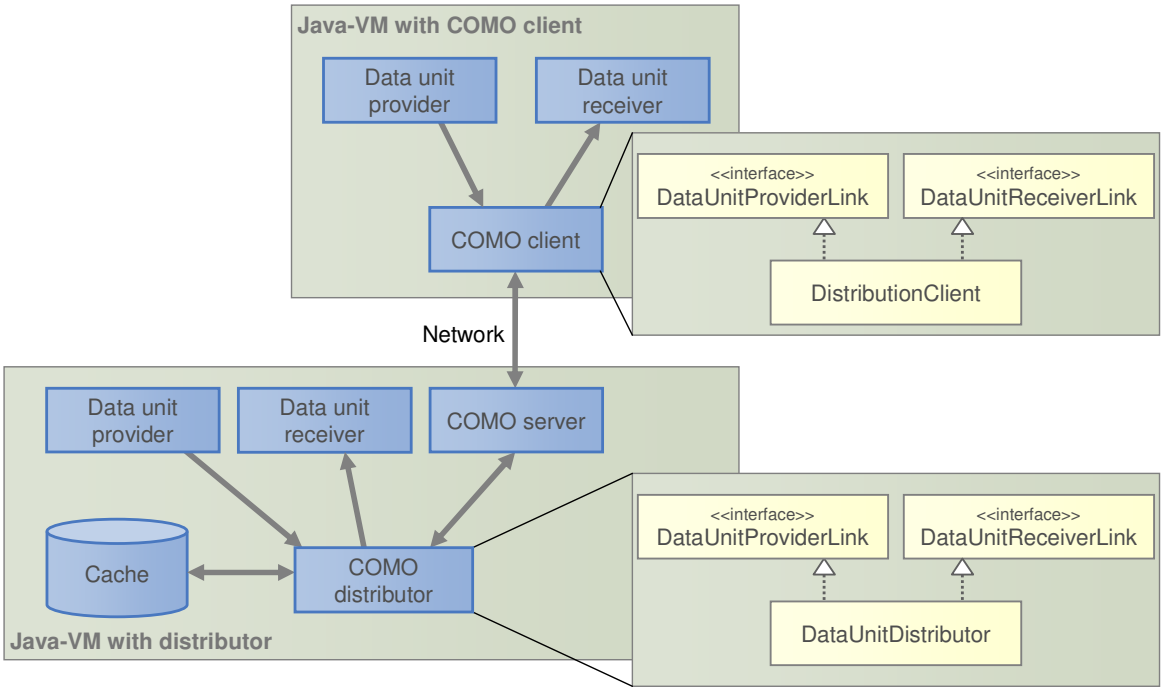

Figure 89: Network Transparency

The COMO client and server will also take care of map matching. Thus locations in the payload of every data unit that is sent over the network will be encoded to AGORA-C to achieve map independent geo-referencing on the sender side and the receiving side will decode the AGORA-C and match it on the local map. This is done by using the POMA AGORA-C en- and decoder services.

Broadcast Use Case – EFCD example

The following section describes the realisation of a COMO broadcast scenario taking the example of the COMO FCD Event generation and provisioning.

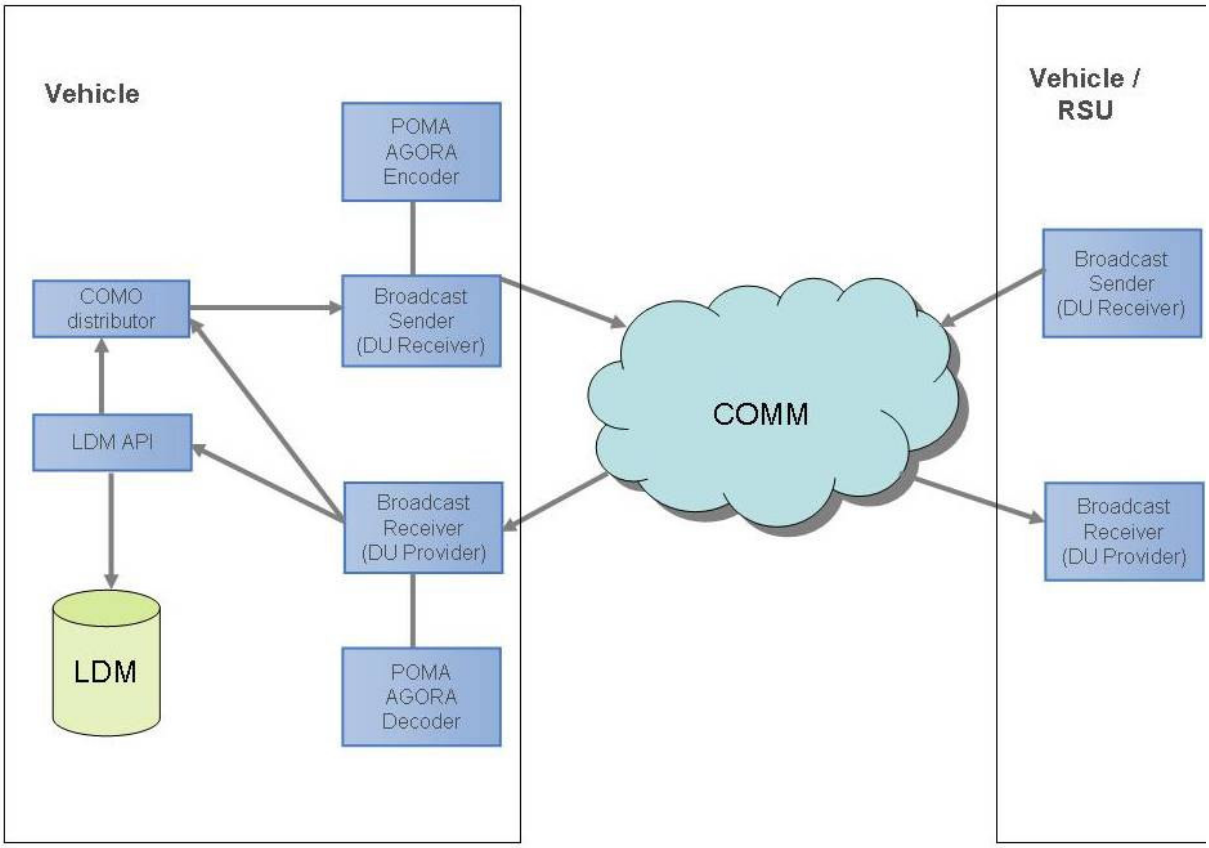

Figure 90: FCD Event in a Broadcast scenario

Sender

1. The Data Fusion or Local Traffic State Calculation generates a new FCD Event and inserts it into the LDM by using the LDM Manager
2. The LDM Manager is registered as DU Provider to the COMO API for the DU FcdEvent
3. The ComoBroadcastService is registered as DU Receiver for DU FcdEvent to the COMO API
4. The ComoBroadcastService receives the new Fcd Event, calls the POMA AGORA-C encode functionality
5. The ComoBroadcastService encodes (streamline information for optimised communication) the Fcd Event object and provides it as Broadcast Content to the appropriate FOAM interface

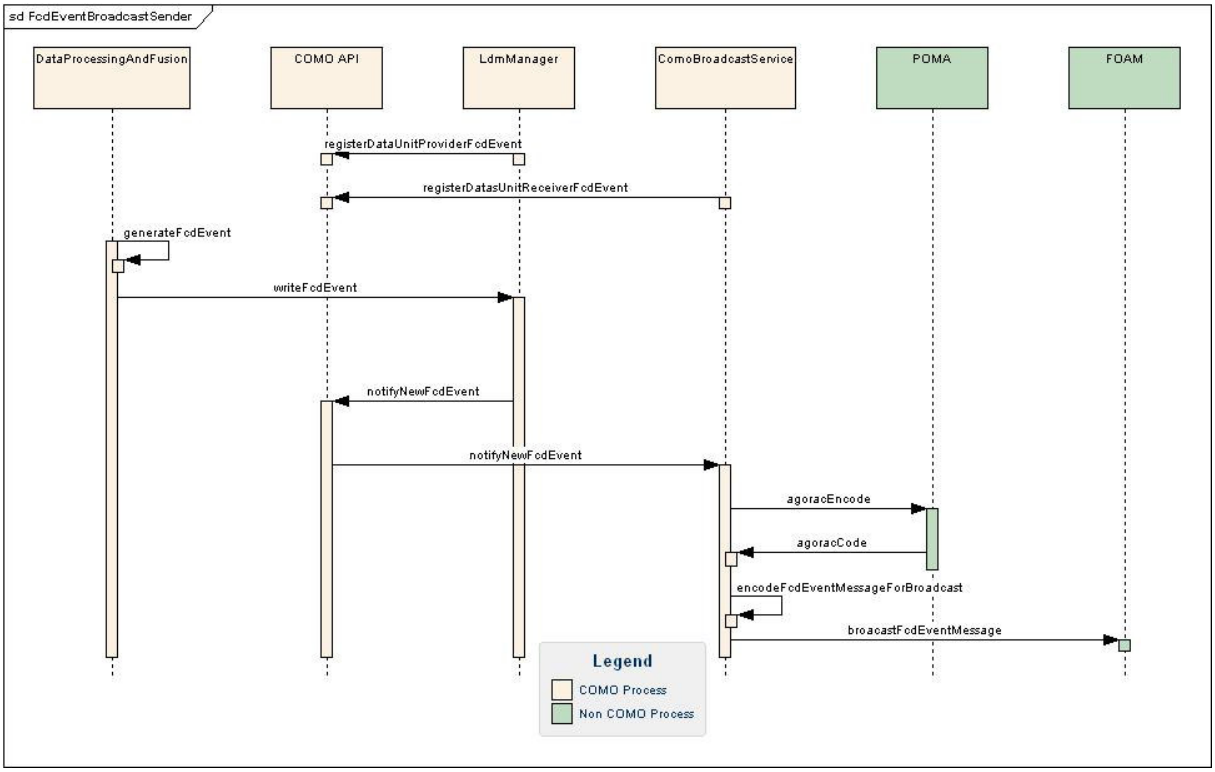

Figure 91: Sequence diagram - FcdEventGeneration and message sender

Receiver

1. The ComoBroadcastService receives the Fcd Event information via the appropriate FOAM CALM interface
2. The ComoBroadcastService module decodes the Fcd Event information and builds a new FcdEvent object
3. The ComoBroadcastService module calls the AGORA-C decode method and inserts the result into the appropriate FcdEvent attribute
4. The ComoBroadcastService module is registered as DU Provider to the COMO API for the DU FcdEvent
5. The ComoBroadcastService provides the FcdEvent to the COMO API
6. All modules registered as DU receive new FcdEvent from COMO Distributor

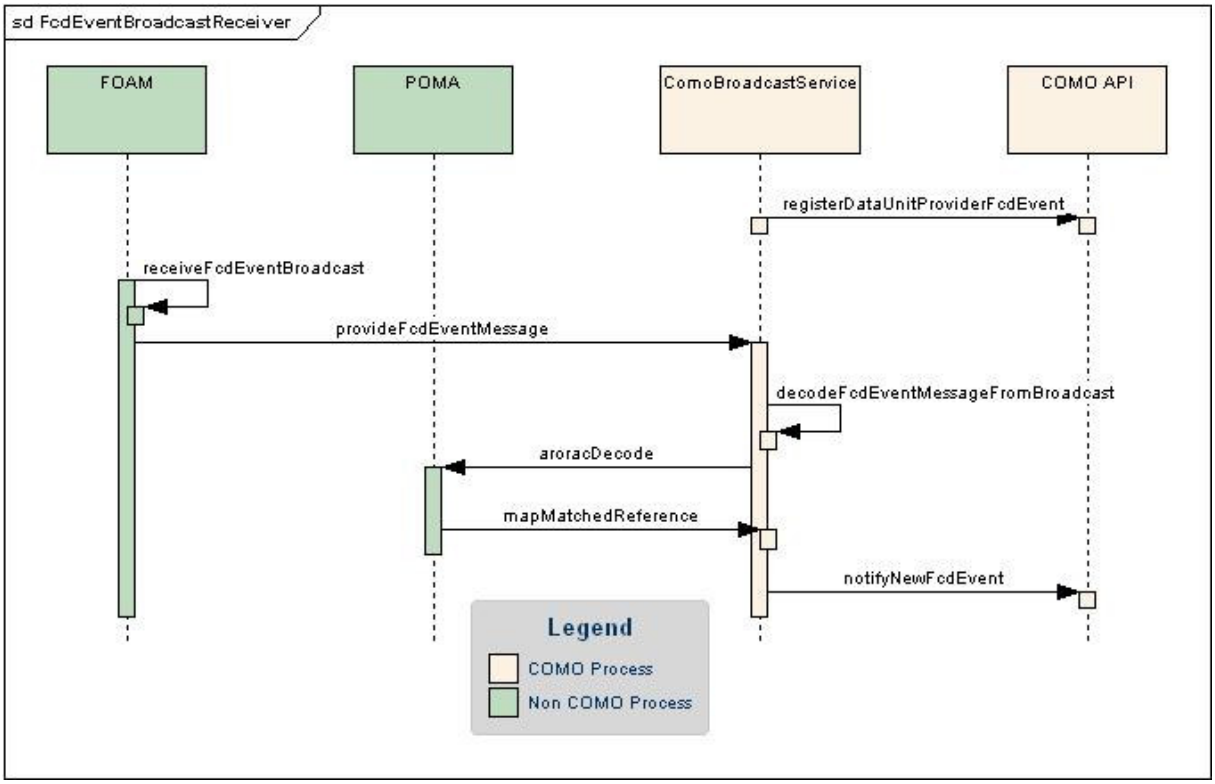

Figure 92: Sequence diagram - FcdEventGeneration and message receiver

COMO API interfaces:

1. to LDM Manager - to realize the scanning of the data base by means of available querying mechanisms
2. to application / service - to receive rules from the applications and send notification messages to the applications

4.6.5 High level composite architecture

The technological objectives for "CoOperative MOnitoring" (COMO) are to provide reliable traffic information and traffic data (vehicle data and messages, local traffic overviews of the RSUs and centre-level traffic overviews) to CVIS applications which may use them to provide their services.

The following figure provides an overview on the COMO components on a high level.

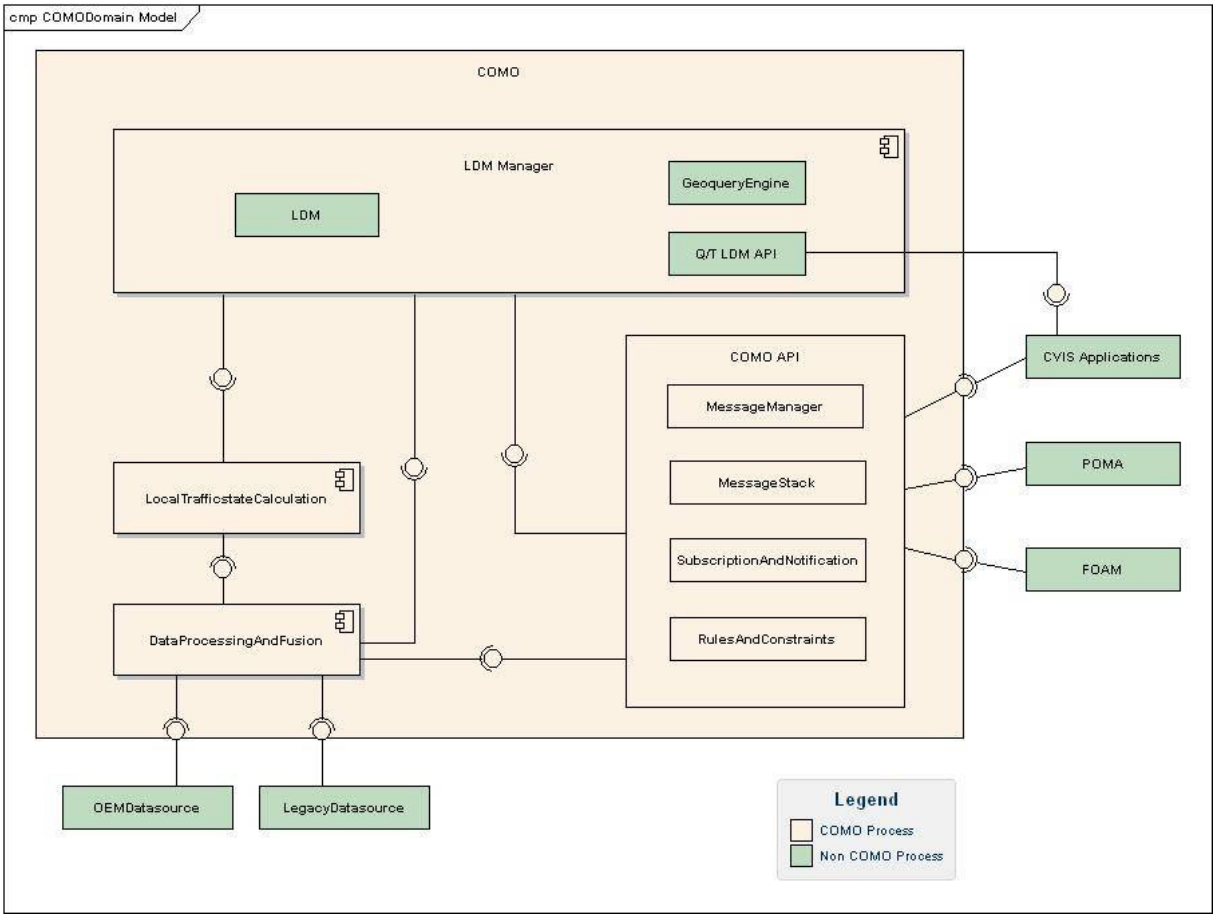

Figure 93: COMO composite diagram

The COMO composite diagram above describes the structure of the COMO sub-system together with its embedding in the overall CVIS system. The elements marked in green are not provided by COMO. The COMO system mainly comprises the following functional units:

1. OEM and Legacy Gateway
   These are the interfaces towards OEM platform to integrate data from the vehicle or the infrastructure side and towards legacy systems to integrate relevant information from traffic management centres.
2. COMO Data Fusion and Traffic State calculation
   COMO Traffic State Calculation and Data Fusion are necessary to provide consistent basic traffic related as well as traffic state information to CVIS applications. These processes access the POMA functionalities for geo referencing of data provided by RSU or vehicle via OEM Gateway which are not geo referenced by nature and map matching of data sets provided by other vehicles through COMO API which are already geo referenced. The COMO Data Fusion and Traffic State calculation components are the only one eligible to write data into the LDM concerning CVIS.
3. COMO API
   This is the interface towards the Local Dynamic Map (LDM) which contains the COMO data. The COMO API thus represents the LDM in CVIS. The COMO API provides access for all CVIS SPs to information provided by COMO. It realises functionalities which are responsible for the exchange of different kinds of information between applications on

different hosts like vehicles, roadside units or service centres, it takes care about subscription for information and message handling and management and it provides a set of rules and contraints to access specific sets of information.

5 Execution infrastructure

This section is based on the D.FOAM.3.2 specification document. A brief overview of the execution infrastructure is provided in the following. For more elaborated specifications of the execution infrastructure we refer to D.FOAM.3.2 and to generally available OSGi and Java documentation.

5.1 Overview

The CVIS execution infrastructure is based on the OSGi framework. Since the current OSGi specification only provides a binding to the Java platform, the execution infrastructure is implemented by a set of standard Java APIs, as well as a "Java Virtual Machine" (JVM), which will run on top of the Linux operating system that comes with the CVIS host

Please note that the dependency from the execution infrastructure on the underlying operating system constitutes a formal interface between the middleware and the communication infrastructure, where the communication infrastructure is the provider of the CVIS host hardware, including the Linux operating system.

5.2 High level composite architecture

The OSGi framework can be divided into the five layers.

1. Security layer
2. Module layer
3. Lifecycle layer
4. Service layer
5. Actual services

This layering is depicted in Figure 94.

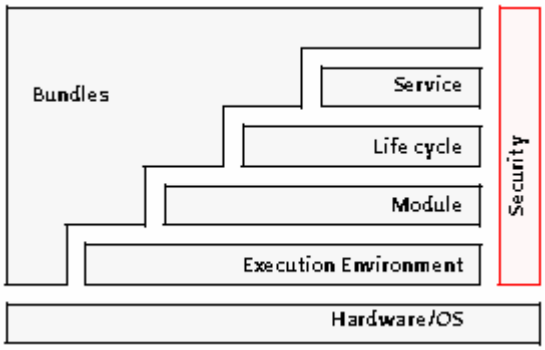

Figure 94: Execution environment parts

These layers provide the necessary functionality to deploy and operate Java bundles, including bundle registering, local interaction (within same CVIS host), lifecycle management etc. Recall that all the CVIS facilities and applications are Java bundles except for native applications (native applications may expose their services in the OSGi based execution infrastructure though).

The *security layer* is based on Java 2 security but adds a number of constraints and fills in some of the blanks that standard Java leaves open. It defines a secure packaging format as well as the runtime interaction with the Java 2 security layer. In CVIS we have specified a more extensive security framework to support security requirements of the ITS environment (see section 3.3).

The *module layer* defines a modularization model for Java. It addresses some of the shortcomings of Java's deployment model. The modularization layer has strict rules for sharing Java packages between bundles or hiding packages from other bundles. The module layer can be used without the lifecycle and service layer. The lifecycle layer provides an API to manage the bundles in the module layer, while the service layer provides a communication model for the bundles.

The *lifecycle layer* provides a lifecycle API to bundles. This API provides a runtime model for bundles and is used for lifecycle management in CVIS. The lifecycle API is presented in section 3.1 which describes the lifecycle basic facility. It defines how bundles are started and stopped as well as how bundles are installed, updated and uninstalled. Additionally, it provides a comprehensive event API to allow a management bundle to control the operations of the service platform. The lifecycle layer requires the module layer but the security layer is optional.

The *service layer* provides a dynamic, concise and consistent programming model for Java bundle developers, simplifying the development and deployment of service bundles by de-coupling the service's specification (Java interface) from its implementations. This model allows bundle developers to bind to services only using their interface specifications. The selection of a specific implementation, optimized for a specific need or from a specific vendor, can thus be deferred to run-time.

A consistent programming model helps bundle developers cope with scalability issues in many different dimensions. This is critical because the framework is intended to run on a variety of devices whose differing hardware characteristics may affect many aspects of a service implementation. Consistent interfaces insure that the software components can be mixed and matched and still result in stable systems.

The framework allows bundles to select an available implementation at run-time through the framework service registry. Bundles register new services, receive notifications about the state of services, or look up existing services to adapt to the current capabilities of the device. This aspect of the framework makes an installed bundle extensible after deployment: new bundles can be installed for additional features or existing bundles can be modified and updated without requiring the system to be restarted. The interactions between the layers are depicted in Figure 95.

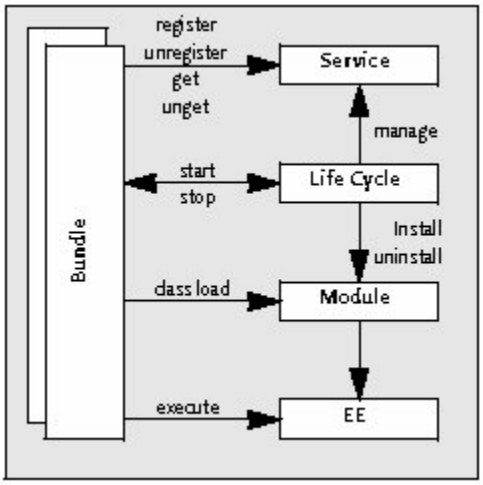

Figure 95: Interactions between layers

5.3 Application programming interface

The lifecycle layer API is specified in Figure 20. The service layer API is specified in Figure 96.

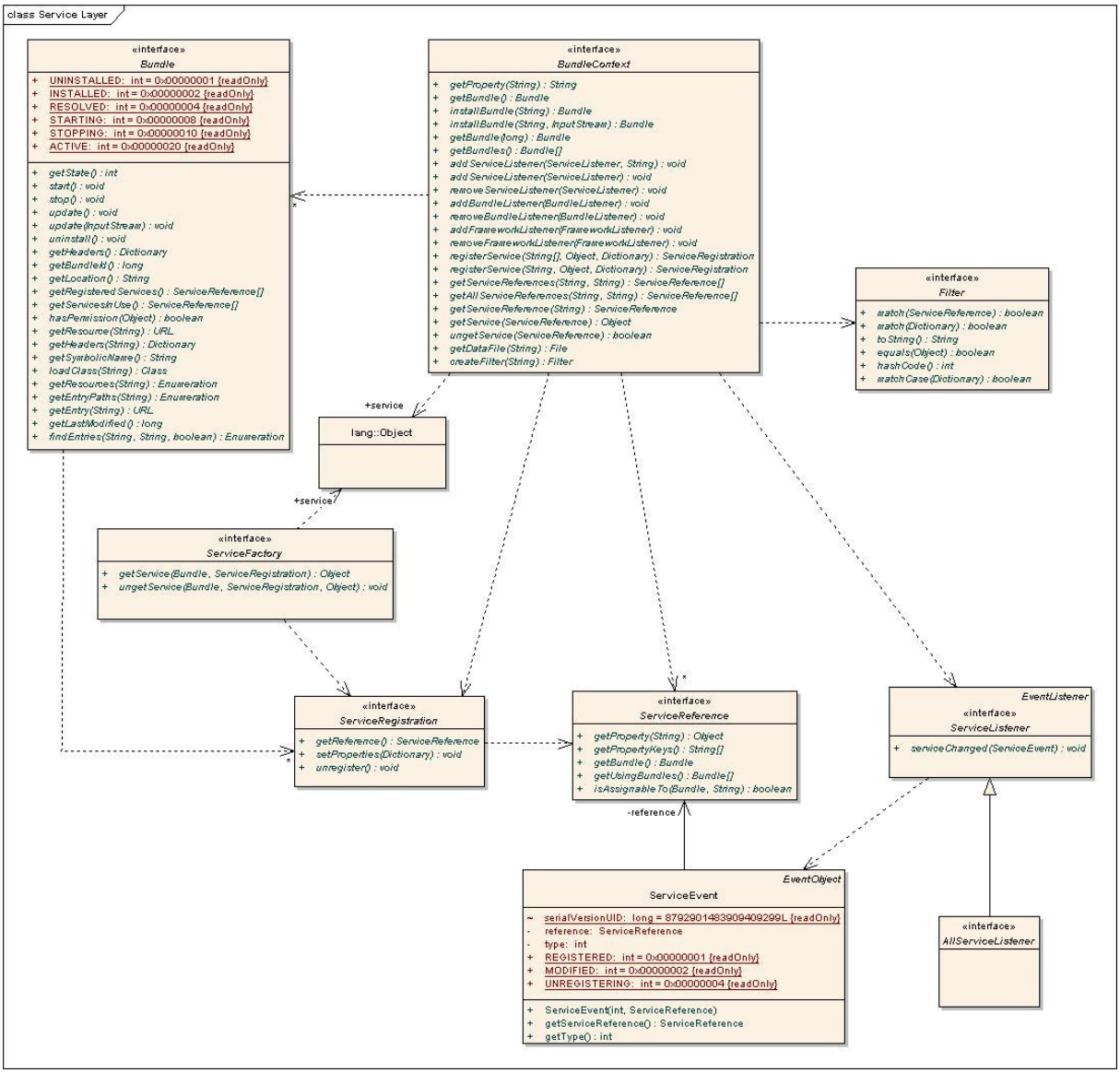

Figure 96: Service layer API

6 Communication Infrastructure

This section is based on the D.COMM.3.2 specification document. An overview of the communication infrastructure is provided in the following. For more elaborated specifications of the communication infrastructure including specifications of hardware (antennas etc) and details of the air interfaces (M5, infrared and 3G) we refer to D.COMM.3.2.

6.1 Overview

The communication infrastructure enables seamless and continuous communication from the vehicle towards the infrastructure and other vehicles. The connection will be transparent to the applications, and will supply socket communications for more demanding applications. Assignment and management of the communication media should be completely beyond the responsibility of the non native applications. However, the communication infrastructure provides only the possibility to communicate with other communication peers. The communication content is within the scope of the facilities and the applications.

The communication infrastructure uses Linux as underlying operating system to provide a flexible and open development environment and to avoid the purchasing of expensive software licenses. Furthermore the communication infrastructure uses the basic set of international CALM ITS communication standards as a basis. These CALM standards are new ISO standards for car-to-car and car-to-infrastructure communication. Most of the CALM standards are finalized. CVIS COMM acted as a proof of concept of draft ISO CALM standards. CALM also provides IPv6 networking in the mobile environment. CVIS is based on IPv6. This decision was taken based on ITS needs and also based on the assumption that IPv6 will take over IPv4. However, both versions of the IP protocol can cohabit, through the use of transition mechanisms. It was implemented and tested within COMM and the results were fed back into the standardization process.

6.1.1 CVIS communication architecture

The communication infrastructure is depicted in Figure 97. It enables host communication through the router and the router air interfaces. It also provides communication to CVIS sub-system internals such as sensors and actuators.

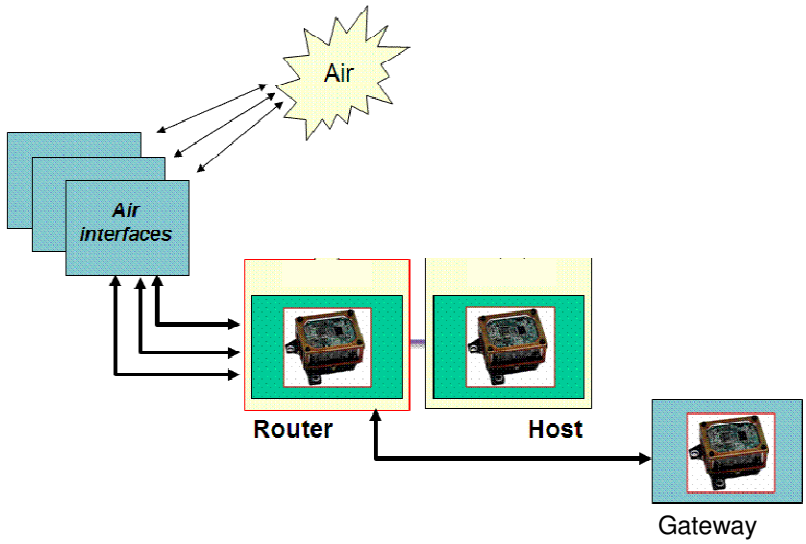

Figure 97: Communication infrastructure

CALM architecture

CVIS will use several communication interfaces (CIs) on the router, to be able to connect to different types of wireless networks. Some of these interfaces are developed by CVIS parties, so they may not have a Linux device driver in the stock operating system. Modification of the stock Linux device driver will be needed even if they have to integrate the interfaces into a CALM system (to make them CALM-aware).

The following air interfaces were used in CVIS:

- CALM M5
- Infrared
- 3G

The CALM based communication stack is depicted in Figure 98. More elaborated descriptions of the communication stack is provided in D.COMM.3.2

A "CALM Device Driver Framework" (CDDF) based on ISO 21218 is provided to interface the communication media and integrate them into the operating system of the router.

The "CALM Device Driver Framework" (CDDF is an in-kernel stack for integrating device drivers. It implements the common device driver parts, and provides the routines and means for the device drivers to be able to communicate with the CALM manager. It only provides the MI-SAP (see ISO 21217) towards the ITS management, and some helper routines for the IN-SAP (see ISO 21217).

Functions provided by the CDDF are:

CI device driver register call-backs in the CDDF. Management messages coming from the IME (after some validity checking and housekeeping) will be passed to these call-back functions.

The means for device drivers to be able to send asynchronous messages about certain

events to the IME.

So the management part of the air interface device drivers (the media management adaptation entity specified in ISO 21218) can be accessed via this framework, and they can communicate with the IME via the means provided by the CDDF.

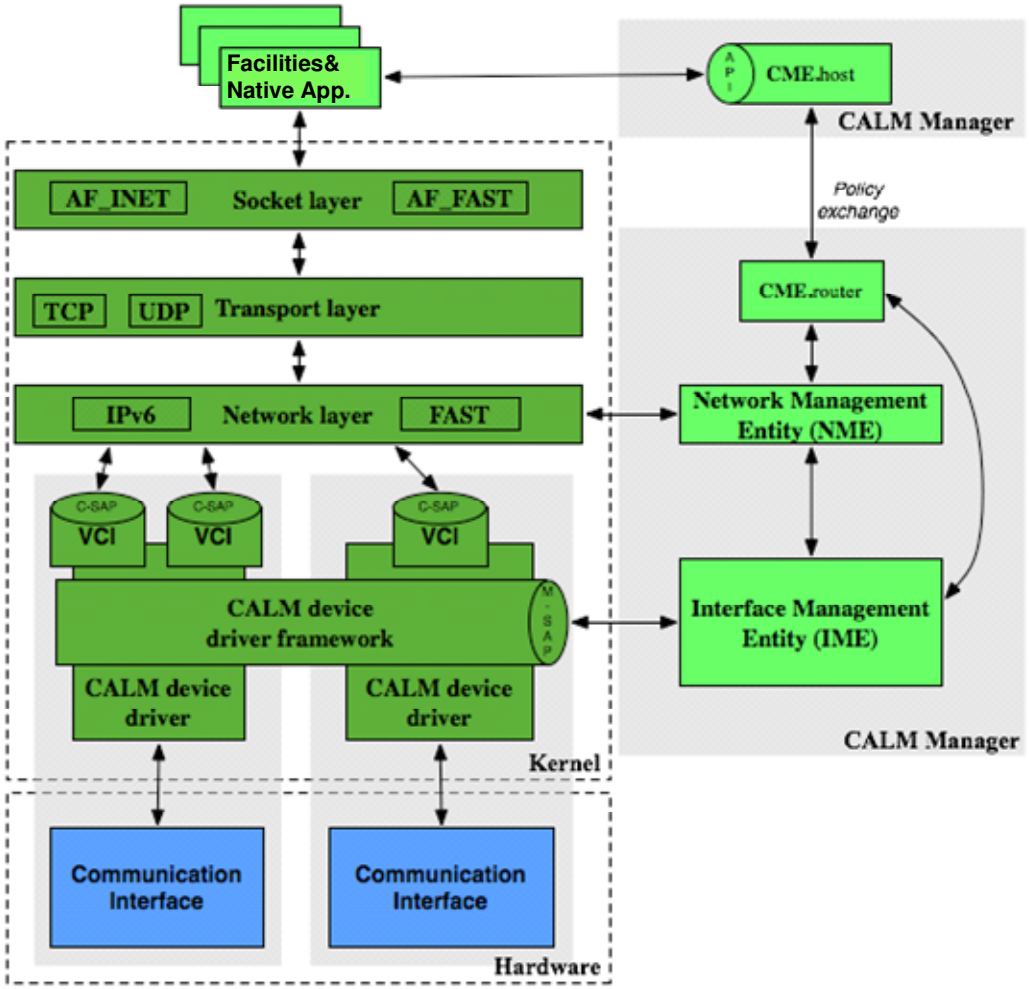

Figure 98: CALM communication stack

6.1.2 CALM ITS standards

These CALM standards are finished and will be published in a first version in 2010. The CALM communication architecture is specified in ISO 21217, based on the ITS station reference model presented below. "Access", "Networking & Transport" and "Facilities" denote the ISO communication layers 1 and 2, 3 and 4, 5 through 7, respectively. "Management" denotes the CALM ITS station and communication management. "Security" denotes the CALM ITS security entity. "Applications" denotes the ITS applications.

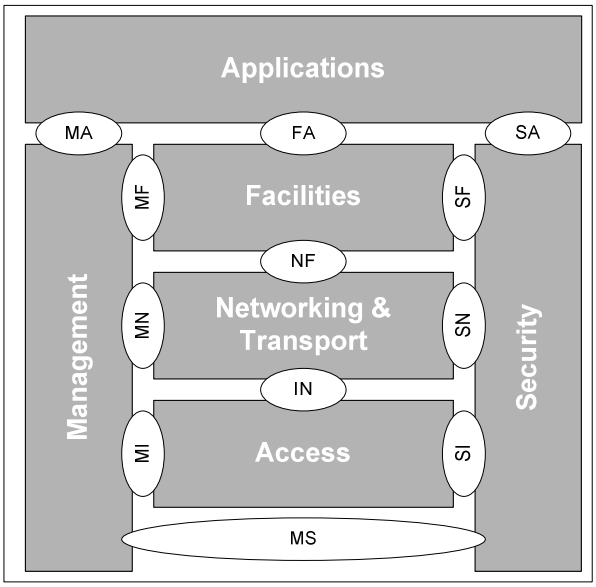

Figure 99: ITS station reference architecture

Access layer

CVIS uses several communication interfaces (CIs) on the router, to be able to connect to different types of wireless networks. Some of these interfaces are developed by CVIS parties, so they may not have a Linux device driver in the stock operating system. Modification of the stock Linux device driver will be needed even if they have to integrate the interfaces into a CALM system (to make them CALM-aware).

At least the following air interfaces were used in CVIS:

CALM M5 (ISO 21215)

Infrared (ISO 21214)

3G (ISO 21213)

Networking & transport layer

CALM introduced IPv6 networking (ISO 21210) for cooperative ITS, which is used in CVIS. This decision was taken based on ITS needs, and is based on the assumption that IPv6 will take over IPv4. However, both versions of the IP protocol can cohabit, through the use of some transition mechanisms.

Additionally, CVIS supports the CALM FAST networking & transport protocol (ISO 29281), which was developed in a joint approach of CALM and CVIS.

Facilities layer

CALM introduced protocol means (ISO 29281, ISO 24102)

- for support of legacy systems (CEN DSRC),
- for service advertisement (similar to WAVE and CEN DSRC) using CALM FAST,
- fpr advertisement via CALM FAST of IPv6-based services (IPv6 prefix announcement),
- for automatic mapping of ITS applications on CIs

which were supported by CVIS

Management

The CALM management functionality specified in ISO 24102 is available in CVIS. This includes also an approach for co-existence of CALM M5 with CEN DSRC in order to protect payment transactions in electronic fee collections systems.

6.1.3 Main use cases and system boundary

The following figure shows the main use cases of the communication infrastructure. The main actor is CVIS host. The CVIS host can be located in either of the CVIS sub-systems (vehicle, road-side, central or handheld). Messages can either be transmitted in unicast or in multicast mode. Unicast communication refers the point-to-point communication between two dedicated communication peers. The communication described in the following picture is always initiated by one actor and can be followed by a response of the communication peer. Periodic sending of message can be seen as a successive response to a requested service.

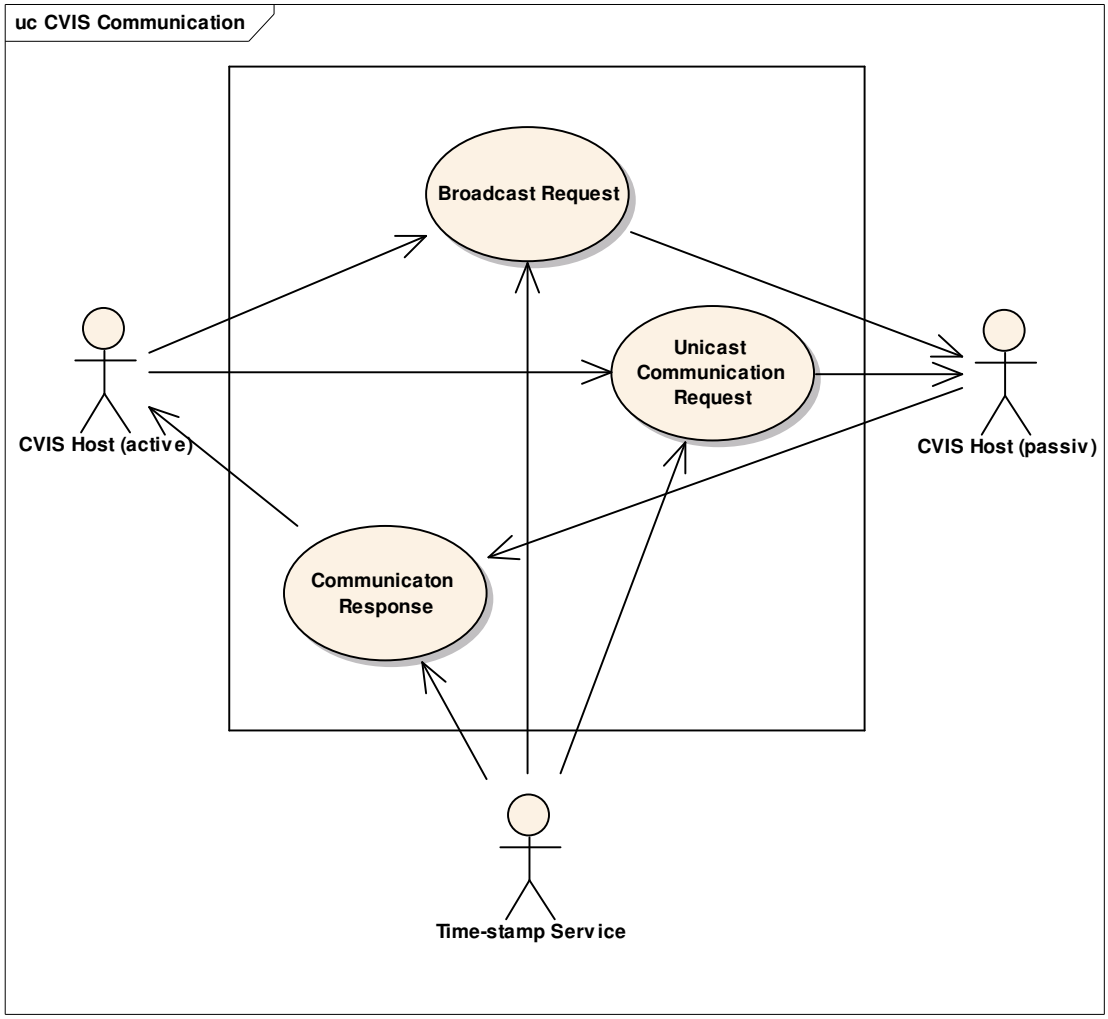

Figure 100: CVIS communication

A broadcast transmits a message to all communication peers within the communication area. Broadcasts will be used e.g. for safety related beaconing and service announcement. An

additional actor outside of the communication infrastructure is the time-stamp service provided by the POMA sub-project. This service adds to the received data packets a time-stamp.

6.2 Domain process model

The main processes of the communication infrastructure are:

Management data.

Transmit data.

Processing.

Receive data.

Time-stamp.

Figure 101 shows the overall domain process model of the communication infrastructure.

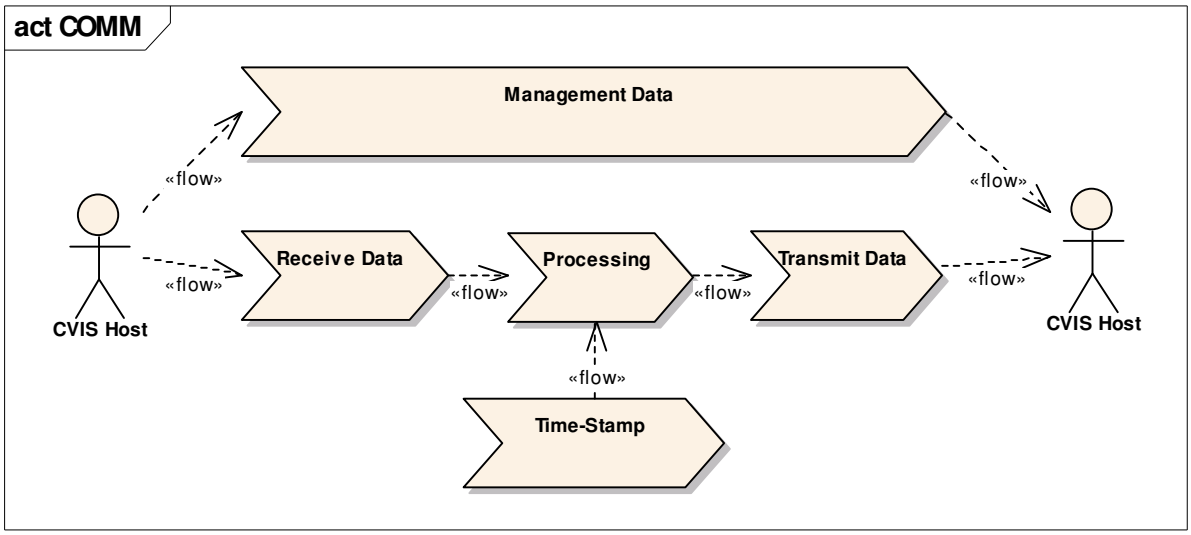

Figure 101: Detailed domain process model

6.3 High level composite architecture

The communication infrastructure can roughly be divided into three communication components and one management component. The three communication components are the three used communication media within the CVIS project:

Cellular communication (2G/3G).

Microwave communication (M5).

Infrared communication (IR).

Note that millimetre-wave communications (MM) was also investigated in CVIS, but not implemented.

The management component contains the management parts of the CALM standards ISO 21218 and ISO 29281. The CALM communication and station management entity specified in ISO 24102 handles the allocation of the communication media.

The three external interfaces of the communication infrastructure are the following:

Management interface: Middleware and applications can register to the communication system and can announce its communication requirements.

Data interface: The transmission of the user data proceeds via this interface. This interface is provided to middleware and applications.

Time-stamp interface: The implementation of the time-stamp service requires an interface to a time-stamp provider, see POMA sub-project.

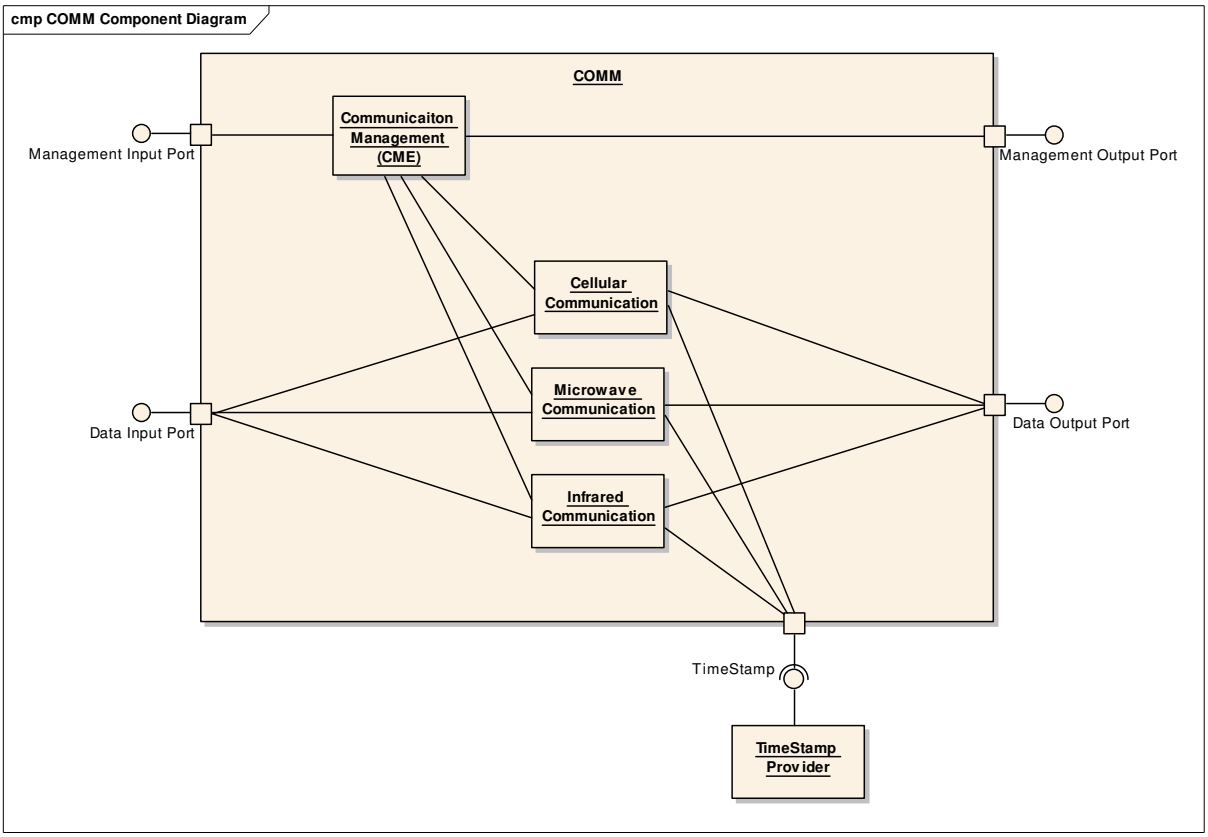

Figure 102: COMM component diagram

The management interface and the data transmission interface are specified in the following sub-sections.

6.4 Management interface

Application programming interface

The management interface provided by COMM is used to access CALM-specific management functionalities, as specified in detail in the appropriate CALM ISO standards. This interface is used by applications which are aware that they are part of a CALM system that can manage a dynamic routing policy based on the needs coming from them. These needs may include data rate, cost of communication, etc.

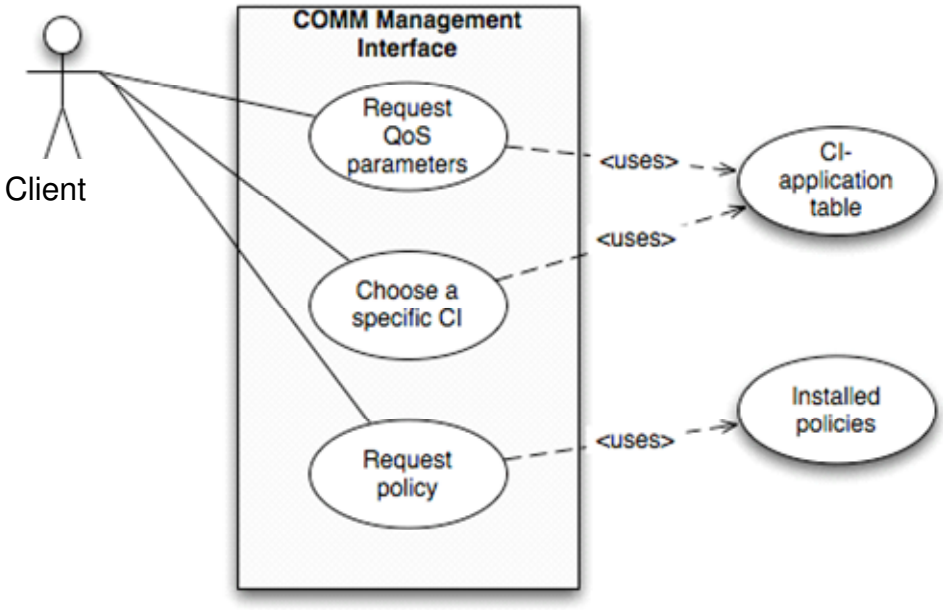

Figure 103: Communication infrastructure management interface

An application may have special "Quality of Service" (QoS) requirements towards the communication sub-system. These special needs are expressed via this management interface. If the networking sub-system can satisfy these needs, then applications will be routed via the appropriate interface chosen.

For instance, an application using some kind of voice over IP protocol may require the networking sub-system to provide a communication channel where the maximum delay and jitter is under some specific value (above which the application can not satisfy user needs, because the quality of the voice would be unacceptable). This application may not be interested in the cost of the communication (but likely this would be a tuneable parameter in a system-wide policy configuration).

Usually, applications are not aware of the exact type and other parameters of the available physical communication devices presented in the system. However, sometimes it may be desirable to give the possibility for applications to directly choose a specific communication device.

The interface described here is the low-level CVIS CALM management interface (which is used to access the CALM management stack of the CVIS system). The API makes it possible to:

Ask the list of currently available CALM-aware communication devices in the system

Set or get the QoS parameters the application uses

Set or get the CALM communication device currently used for communication by the application

Set or get the CALM policy used for the pairing of QoS requirements and available communication devices.

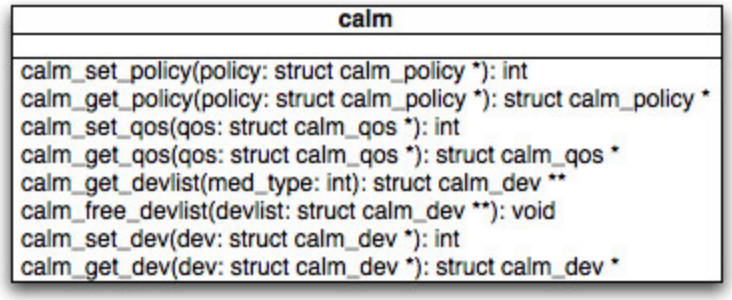

Figure 104: Management API (CALM)

Information model

The main data structures of the management interface which are CALM based are:

struct calm_dev.

struct calm_qos.

struct calm_policy.

The first one, struct calm_dev represents a CALM-aware physical communication device. Applications can choose a device directly if they wish, using calm_set_dev(), and ask for the device used for their communication channel using calm_get_dev().

The list of available communication devices can be asked using calm_get_devlist(), which takes as a parameter the type of the devices the application wants to get. All CALM communication devices can be asked using CALM_MED_UNSPEC as the parameter. The list must be explicitly freed using calm_free_devlist().

A special QoS for a communication channel can be requested using the struct calm_qos data structure. An application can ask the currently used QoS profile using calm_get_qos(), and set a new one using calm_set_qos().

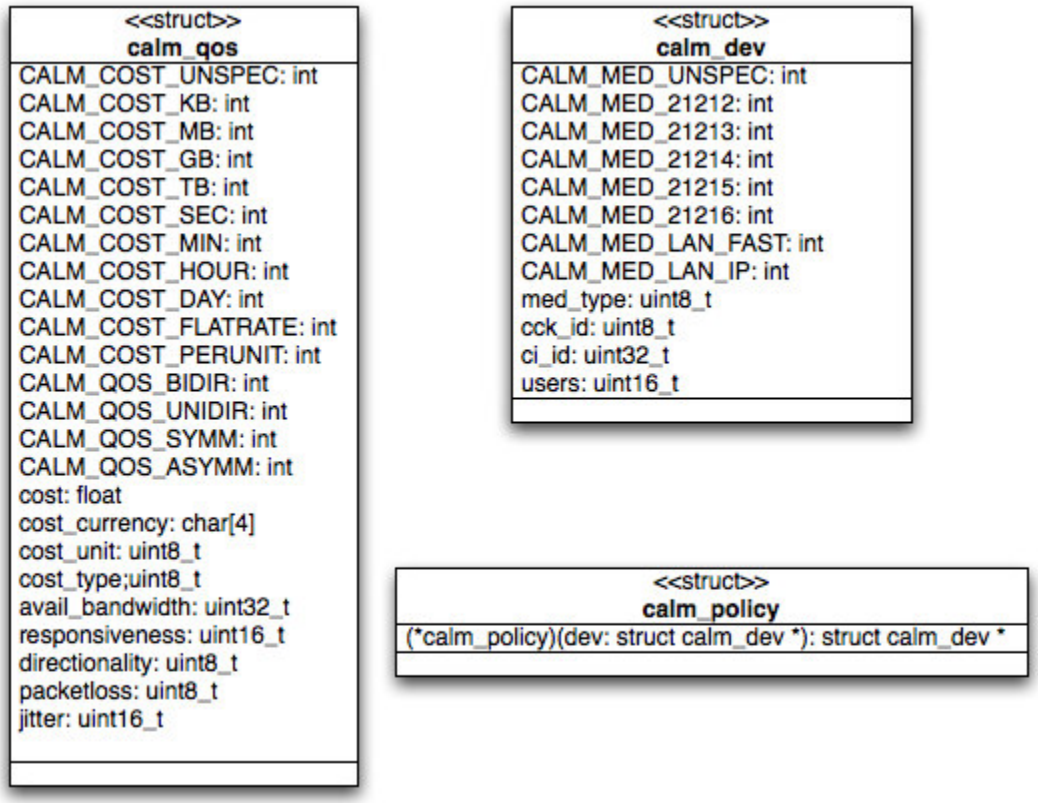

Figure 105: CALM data structures

The pairing process between applications and available communication devices is done using some policy, which by default is set to be system-wide by the administrator. This policy is represented by the struct calm_policy data structure. This can be influenced by an application using calm_set_policy(). There can be several policies installed in a CVIS system, and applications can implement their own one, after which this new policy can be chosen using calm_set_policy() in the same way. The currently used policy can be asked using calm_get_policy().

Interaction model

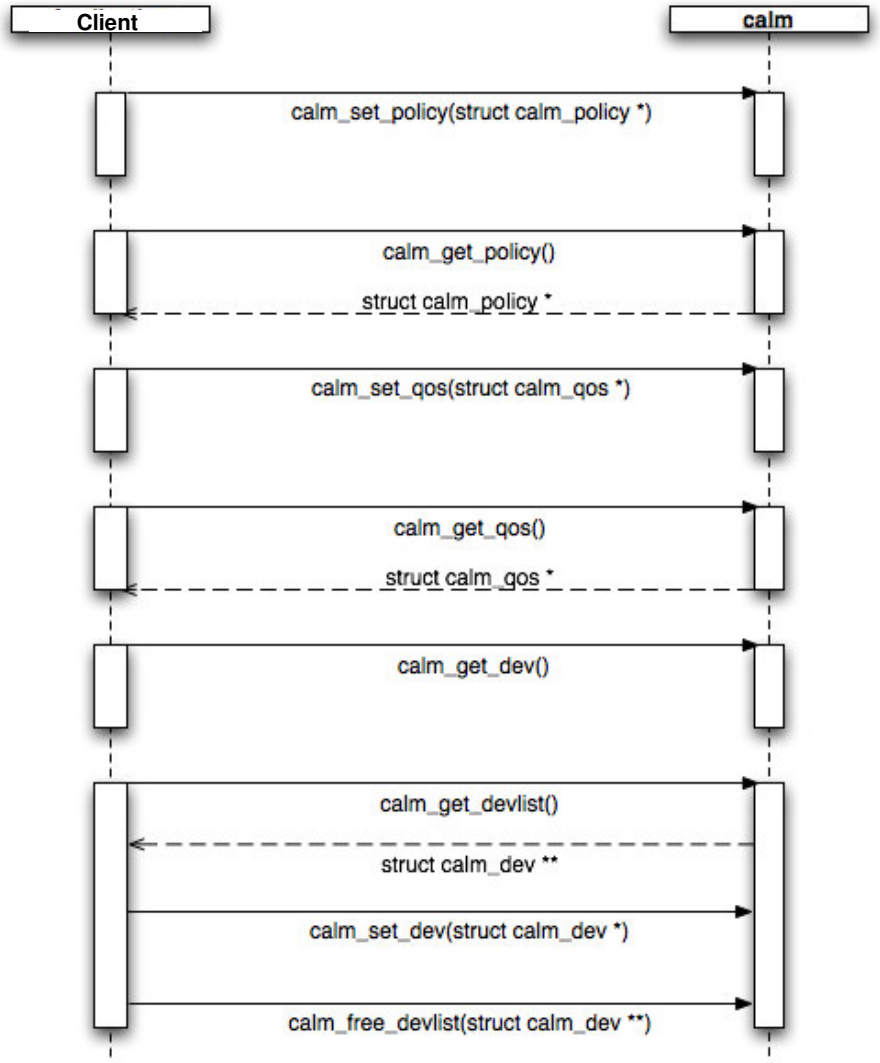

Figure 106: Management sequence

6.5 Data transmission interface

The data transmission interface provided by the communication infrastructure is the usual Linux IPv4/IPv6 socket interface. It is the point where one can access the communication services provided by the communication sub-system.

Applications can be native or Java applications. The interface described here is the low-level data transmission interface (which is used to access the socket layer of the networking stack). It can be directly used by native applications. The JVM implements the necessary classes for Java applications to access the data transmission interface (which are not described here).

Applications can be client or server applications, or they can incorporate functionality for both. Both client and server applications use this interface to

Initiate an outgoing connection

Wait for incoming connections

Send or receive messages

Tune connection parameters.

Initiating an outgoing connection is usually done by client-side applications. It is used e.g. by web browsers to access a web server. In contrast, server-side applications usually wait for clients to connect, and then serve the needs of the latter, e.g. sending an HTML page. An application can implement both functionalities, e.g. peer-to-peer applications, which act both as servers and clients at the same time.

An application can send or receive messages via an open communication channel, which is the most usual transaction done using the data transmission interface.

To request special connection parameters, an application can influence some kind of attributes of the communication channel. These include the sending and receiving buffer sizes in the kernel, requesting non-blocking communication mode, etc.

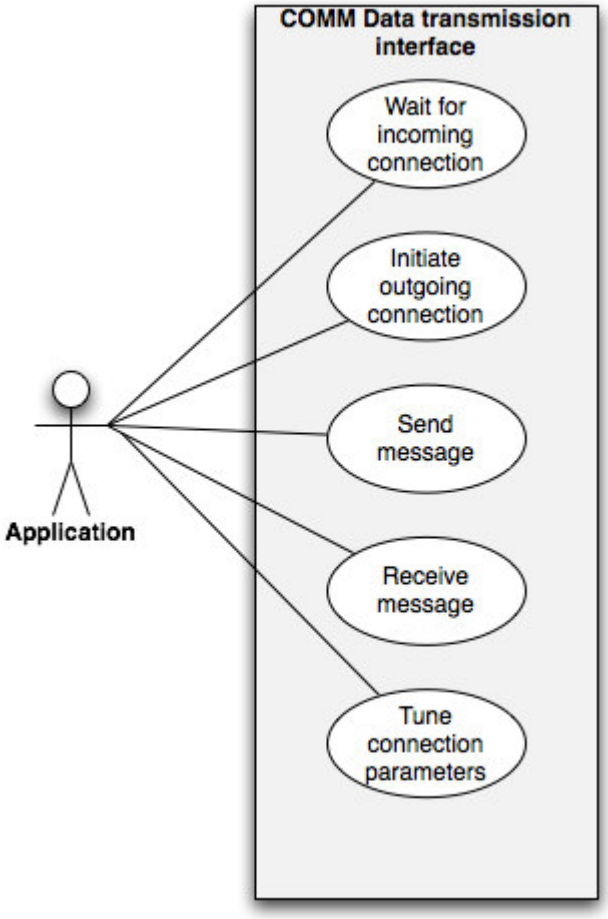

Figure 107: Communication infrastructure data transmission interface

Application programming interface

The API to access the networking sub-system is the well-known BSD socket API, as implemented on Linux. The API described here is the low-level BSD socket API provided by GNU libc.

Applications should not care about the underlying networking stack, or how their message reaches its destination. The complex networking sub-system of CVIS should be totally transparent for them.

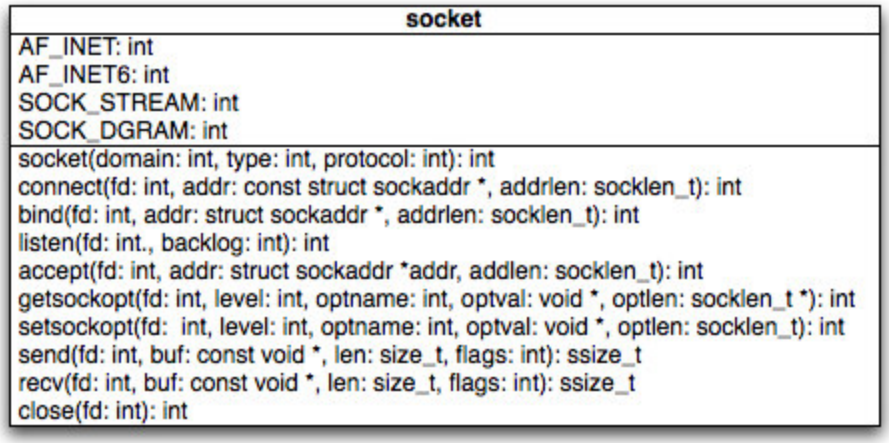

Figure 108: Data API

It is very important to write applications being as agnostic with regard to the underlying network layer protocol as possible. Neither network protocol-specific routines nor hard-coded IP addresses should be presented in applications to make them as flexible as possible. This is especially important for CVIS, because the large scale of mobility means that applications will have to work in different locations with very different networking topologies and technologies.

Information model

The key concept of the BSD socket API is the socket, which is treated under Linux the same way as a regular file descriptor. It represents a communication endpoint, and can be created via the socket() call. The parameters to the socket call tell the networking sub-system what type of communication scheme the application wants to create:

The AF_INET domain tells that it explicitly wants to use IPv4.

The AF_INET6 domain tells that the application wants to use IPv6.

The SOCK_STREAM type means that a reliable communication channel is wanted.

The SOCK_DGRAM type means that no reliable communication channel is needed. Messages sent by the application can arrive to the destination out of order, or can even be lost.

Another key concept is the sockaddr structure, which represents the address of a socket. The BSD socket API and the Linux kernel in its socket-related system calls use it to pass network and transport layer addresses and other information (flow information, IPv6 scope ID, etc). The protocol-specific socket address structures (struct sockaddr_in, struct sockaddr_in6) are

casted to struct sockaddr in the API calls.

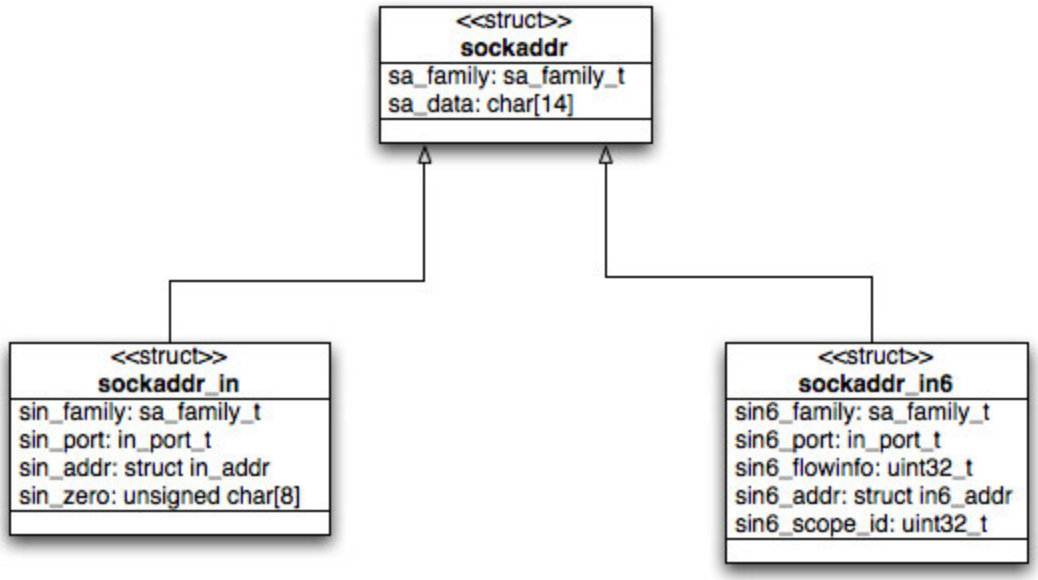

Figure 109: Data socket

The flags parameter is presented in some of the socket functions. It can be:

MSG_OOB: Requests receipt of out-of-band data that would not be received in the normal data stream.

MSG_PEEK: This flag causes the receive operation to return data from the beginning of the receive queue without removing that data from the queue.

MSG_WAITALL: This flag requests that the operation block until the full request is satisfied.

MSG_TRUNC: Return the real length of the packet, even when it was longer than the passed buffer.

MSG_ERRQUEUE: This flag specifies that queued errors should be received from the socket error queue.

Interaction model

The interaction model for the data transmission interface is shown in Figure 110.

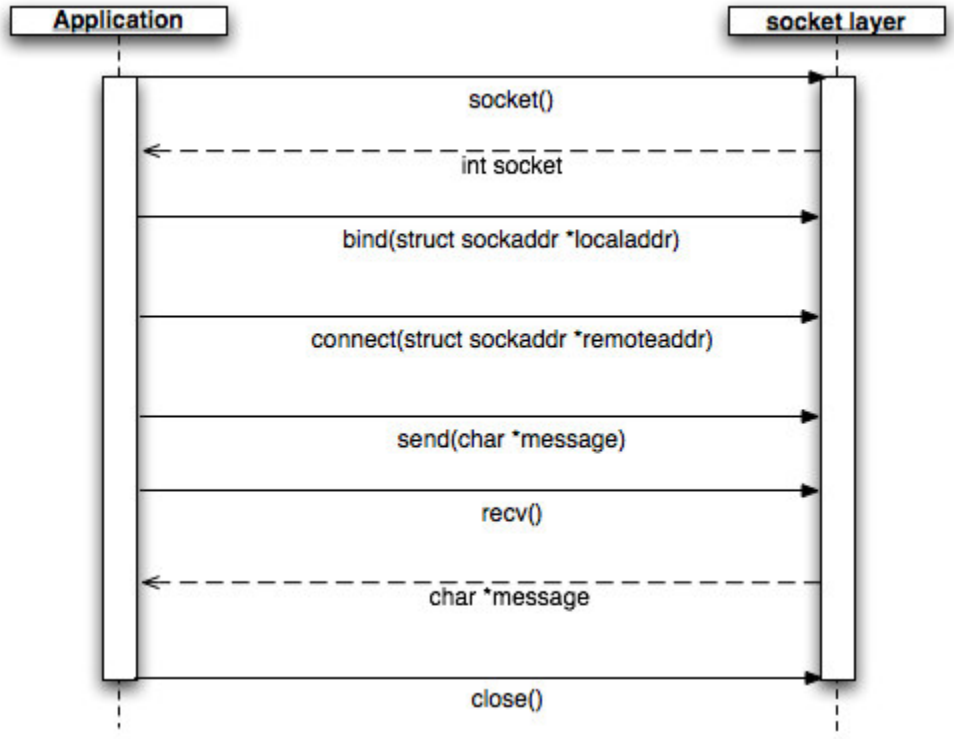

Figure 110: Data sequence

PART III CVIS applications

7 Applications overview

The following set of applications is specified in CVIS:

Dangerous goods; which support transportation of DG through Europe. The focus of the application is on the monitoring and routing of a dangerous goods vehicle during its journey

Parking zones; which provides two main services, one for booking urban parking zones and one for booking highway resting areas.

Access control; which controls access of vehicles into particular regions. The basic ideas of the access control application is to monitor vehicles approaching sensitive zones in order to allow/deny the access, as a preventive safety measure to avoid accidents and as a tool to control dynamically traffic conditions in restricted areas.

Cooperative traveller assistance; which provide assistance to travellers and drivers of other vehicles, e.g. heavy goods vehicles, but not public transport and emergency service vehicles. The assistance to be provided comprises: a) pre-trip and on-trip planning; b) on-trip seamless service with tracking and rerouting if needed; c) vehicle data feeding to traffic control centres.

Enhanced driver awareness; which provides awareness to travellers when they are driving vehicles as part of the journeys and to drivers of other vehicles, e.g. heavy goods vehicles, but not public transport and emergency service vehicles. The facilities provided comprise: a) advice on driving conditions for the part of the road network that is immediately ahead of the vehicle's current position; b) detection, management and provision of advice about ghost drivers.

Information application; which provide subscription services enabling a driver to subscribe to receive information about traffic states, incidents information and alternative route options

Priority application; which provide priority services associated with traffic light controlled intersections

Speed profile application; which provides speed profiles to increase efficiency of an intersection and the traffic network

Cooperative traffic control; which optimize traffic flows in a limited area (up to 5 intersections), based on available traffic information.

Flexible bus lane; which allows a driver within its private vehicle to access a reserved bus lane (BL) using available traffic data such as route guidance and public transport

Network assessment; which provide measuring of the performance of the network. In particular the network assessment is aimed to: i) analyzing the current state of the system, ii) assess the quality of the state of the network, iii) perform an off-line historical analysis, iv) provide ability to compare performance with or without the control system working

Routing application; which provides suggestions of routes that take into consideration the strategy of the network and the willingness of following the route.

Strategy application; which provides design of the traffic management strategy based on traffic status and prediction, traffic demand pattern and statistical assignment.

Traffic control assessment; which aims at assessing the traffic model and to estimate local traffic model parameters. It is meant to be a local application that runs in parallel to the traffic control sub-system and that allow the system to tune the parameters, estimate dynamic parameters and possibly to assess the behaviour of the local traffic controller.

Each of the applications is presented in the next sub-sections applying the following viewpoints:

Overview*;* which provides an overall introduction to the application and its main services

Application programming interface; which describes the logical interface specifying the services the application provides to the end user, e.g. driver, traffic operator etc., via the HMI developed for the application.

Information model; which specifies the application from an information perspective describing information objects of the application domain.

Interaction model; which specifies main usage scenarios associated with the application.

High level composite architecture; which specifies the main components constituting the application.

Deployment model; which specifies the logical deployment of the application, e.g. what parts are deployed on a road-side unit and what parts are deployed on the vehicle.

This document (D.CVIS.3.3) includes specifications of the external interfaces and the overall architecture. Further details as well as the internal design are specified in the corresponding D.SP.3.2 documents.

Note that the application use cases are derived from the requirements of the D.CVIS.2.2 section 4.

7.1 Dangerous goods

The "Dangerous Goods" (DG) application and its main services are introduced in this sub-section. Further details of the DG application can be found in the D.CFF.3.2 "Architecture Specification" document.

7.1.1 Overview

Within the DG application the focus lies on the monitoring and routing of a DG vehicle during its journey. The following figure includes the different use cases. A DG vehicle wants to start its journey and has to register at a traffic management centre which is monitoring and routing the vehicle during its trip. The traffic management centre provides route guidance to the DG vehicle and the vehicle is sending back the information of its position and its status.

The fleet operator and further call centres like police and emergency services have the possibility to have a look at the registered dangerous vehicles by means of a map display.

The traffic supervisor defines the DG vehicle preferred network and in case of an incident he decides on temporary changes of this network to reroute the vehicle in an efficient and safe way.

The subsequent figure shows an overview of the DG application.

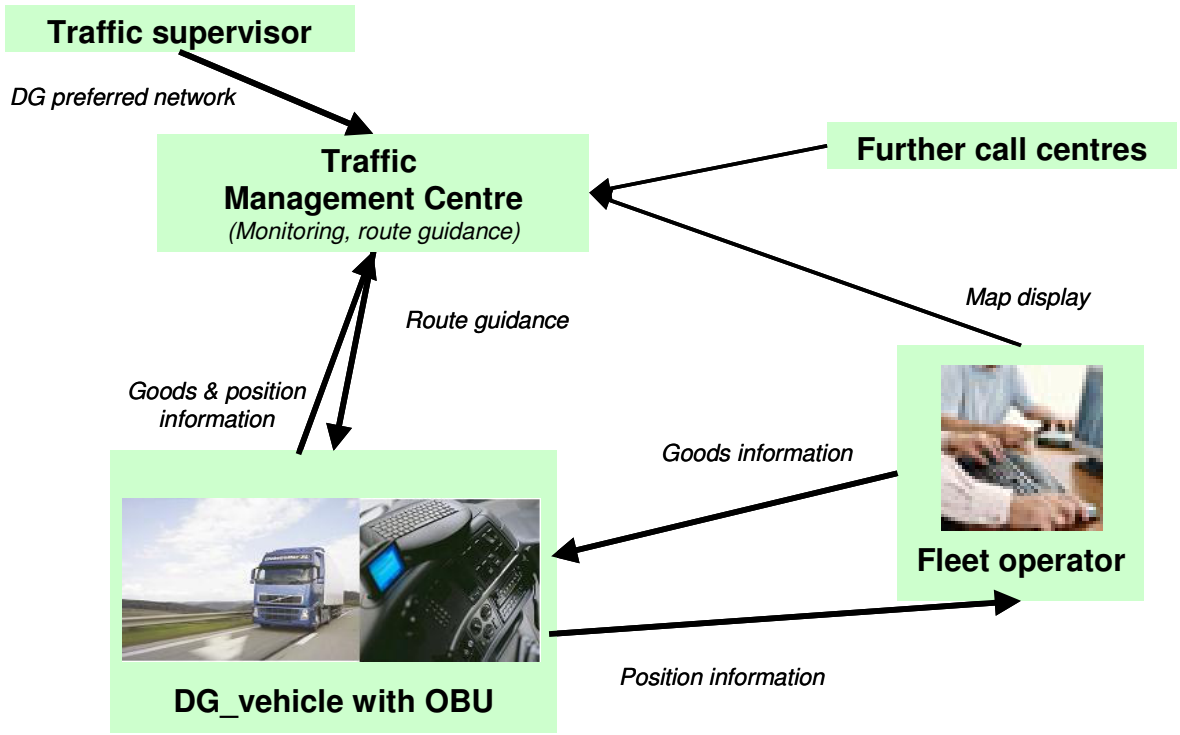

Figure 111 Overview of the DG applications

Main use cases and system boundary

The following diagram shows the involved actors, the use cases and the system boundary of the DG application.

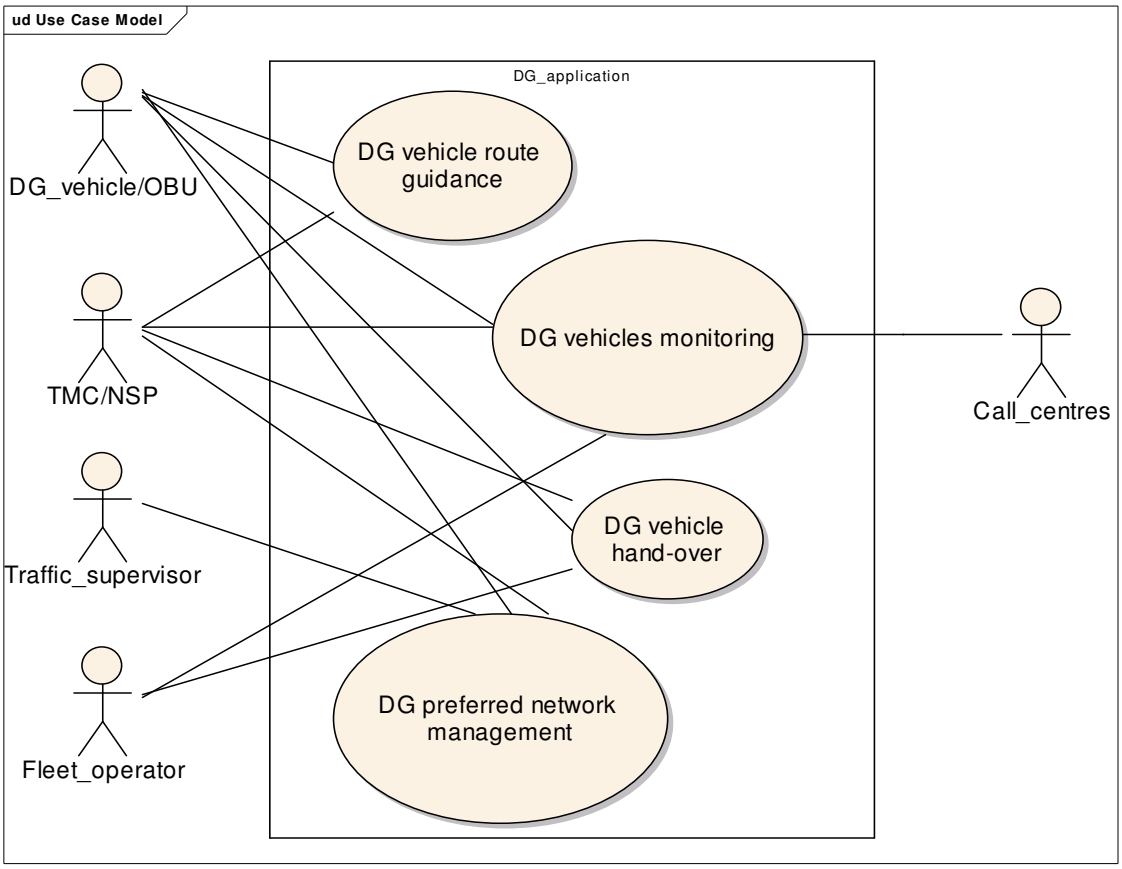

Figure 112: Main use cases and system boundary for the DG application

The DG application contains four use cases:

The route guidance of a DG vehicle including the registration process at a traffic management centre.

The monitoring of a registered DG vehicle from the side of the traffic management centre.

The hand-over of a registered DG vehicle between two traffic management centres.

The navigation and routing of a DG vehicle on its preferred DG network.

The actors of these applications are the following:

DG vehicle/OBU.

Traffic management centre/navigation service provider.

Traffic supervisor.

Fleet operator.

Call-centres.

7.1.2 Application programming interface

The main interfaces and their provided services are depicted in Figure 113.

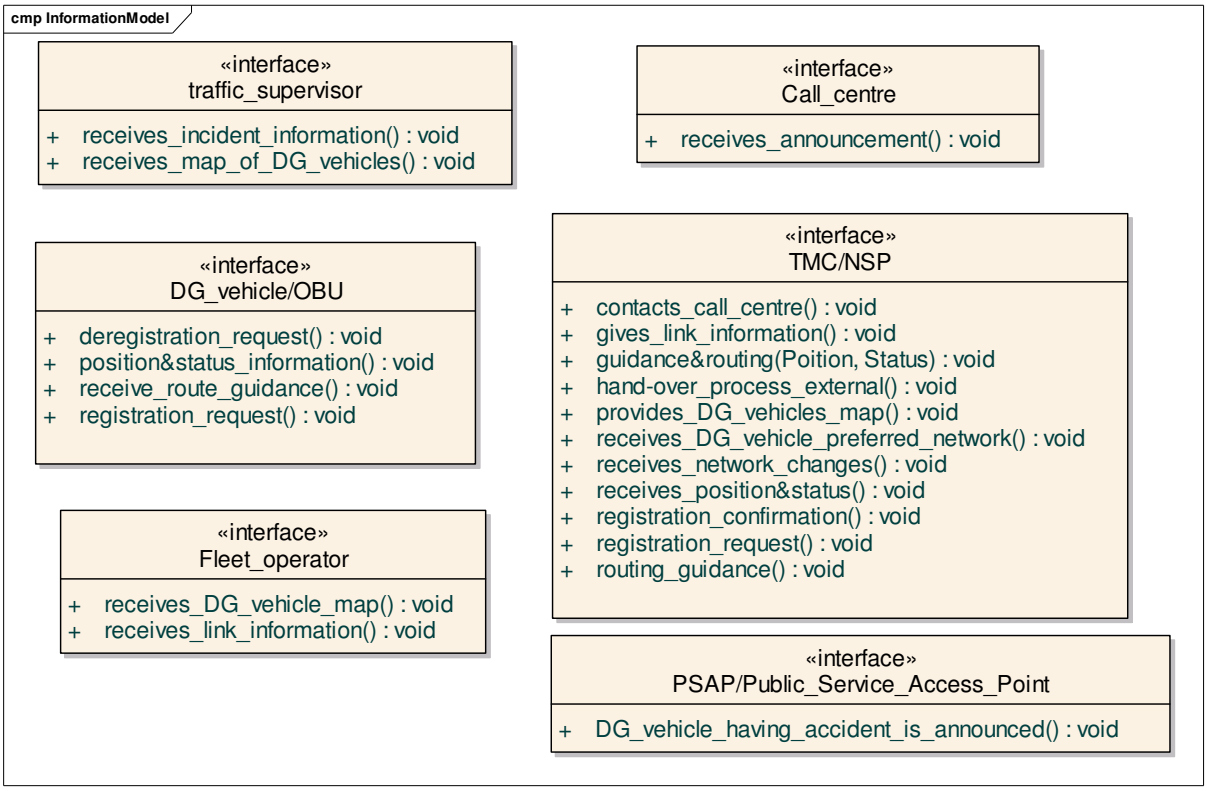

Figure 113: DG information model

An example scenario illustrating usage of the provided interfaces for the *DG vehicle route guidance* use case is shown in Figure 114. The traffic management centre registers the DG vehicle and provides route guidance.

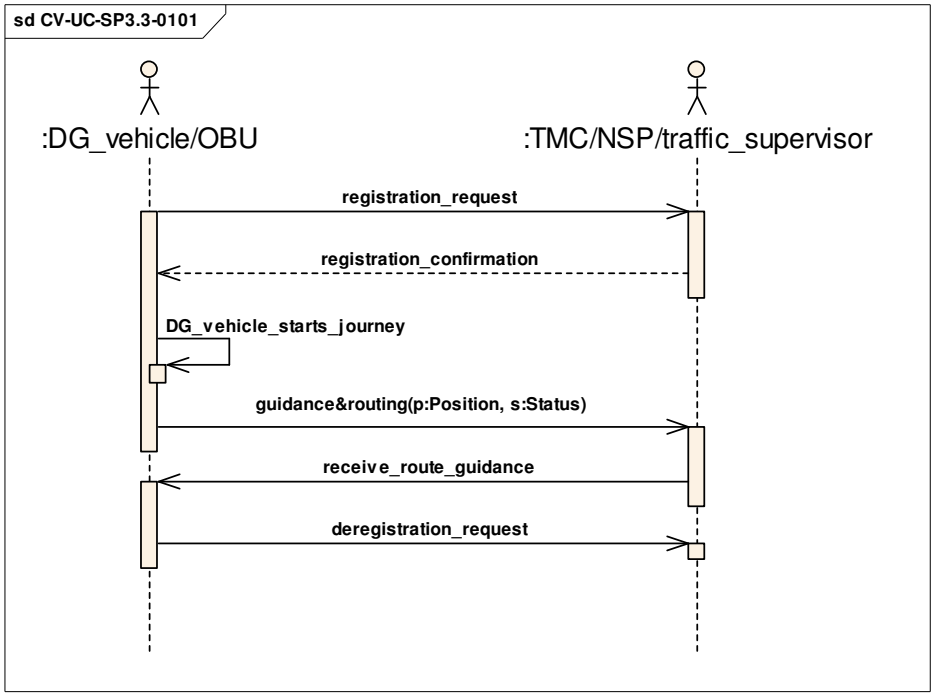

Figure 114: Sequence diagram for the DG vehicle route guidance

7.1.3 Information model

The figure below defines general domain model for the DG.

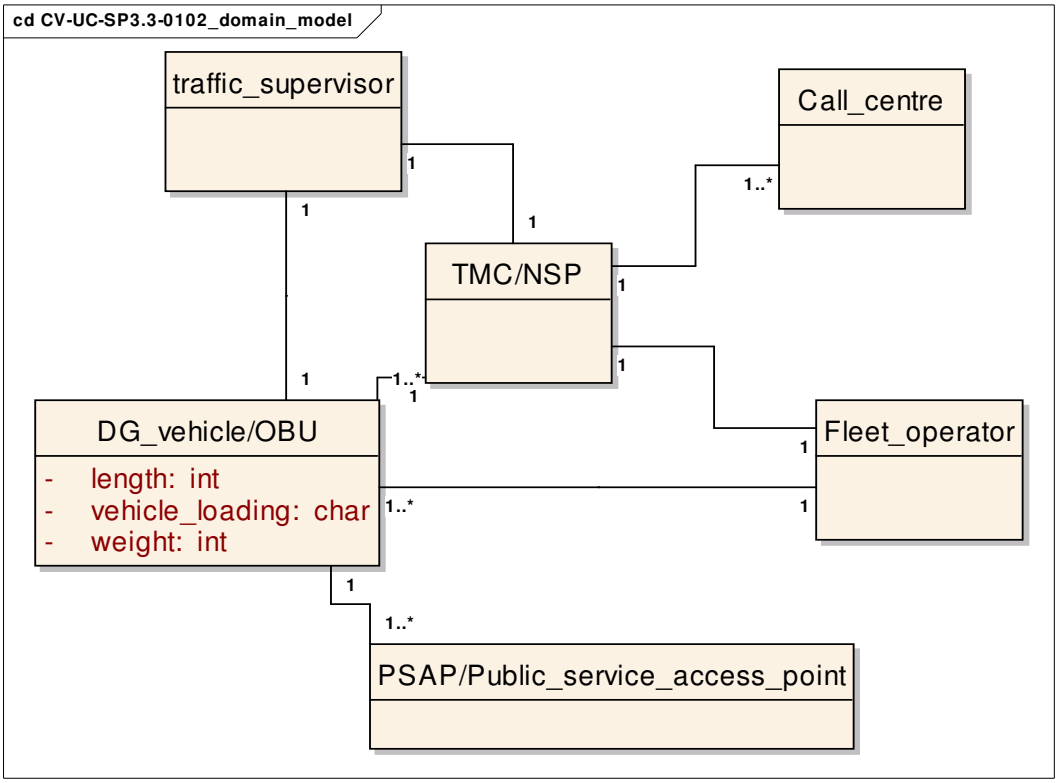

Figure 115: Domain model for the DG vehicles monitoring

7.1.4 Interaction model

The interactions of the four main use cases of the DG application, i.e.

the DG vehicle route guidance,

the DG vehicles monitoring,

the DG vehicle hand-over,

and the DG preferred network management

are specified below using UML activity diagrams.

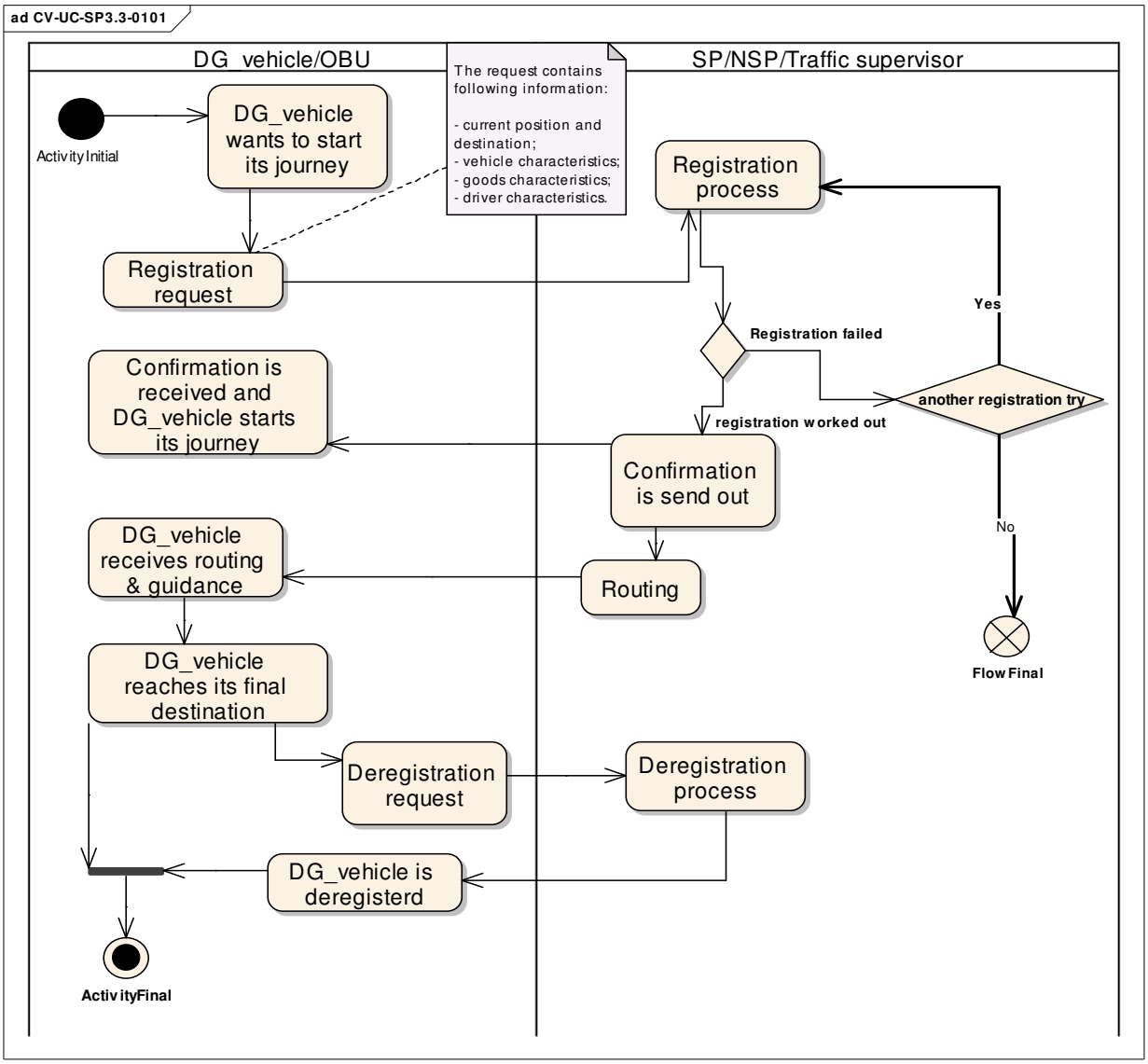

Figure 116: Activity diagram for DG vehicle route guidance

The table below describes the roles identified in the activity diagram and their responsibilities for this particular scenario.

Role	Responsibilities
DG_vehicle/OBU	The DG_vehicle/OBU is responsible to send the registration and the deregistration request to the TMC / NSP / traffic_supervisor. During its trip the DG vehicle is also responsible to send position & status information to the TMC.
TMC/NSP/traffic_supervisor	The TMC / NSP / traffic_supervisor is responsible for the registration/deregistration of the DG vehicle such as the route guidance during its trip.

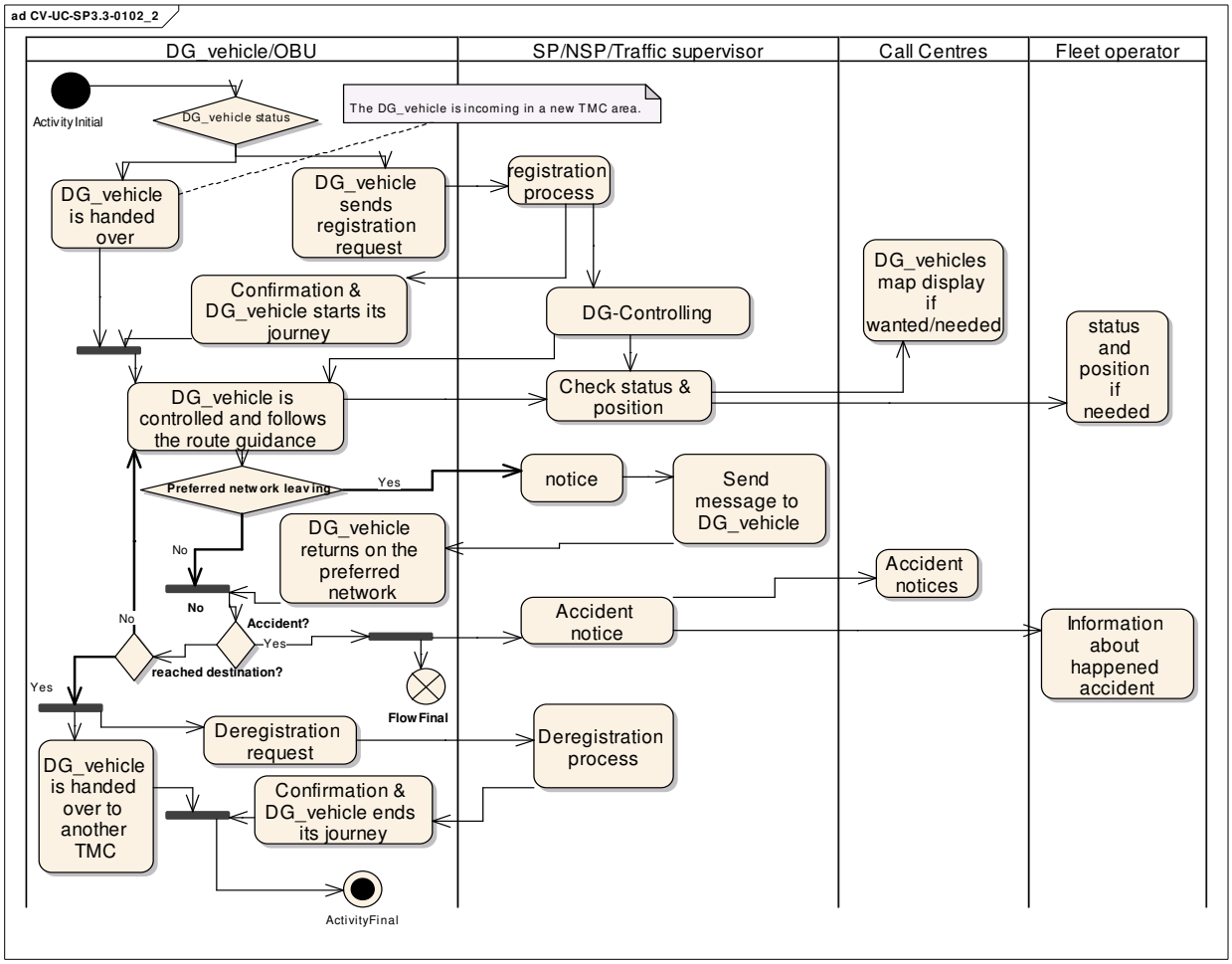

Figure 117: Activity diagram for DG vehicles monitoring

The table below describes the roles identified in the activity diagram and their responsibilities for this particular scenario.

Role	Responsibilities
DG_vehicle/OBU	The DG_vehicle/OBU is responsible to send the registration and the deregistration request to the TMC / NSP / traffic_supervisor. During its trip the DG vehicle is also responsible to send position & status information to the TMC.
TMC/NSP/traffic_supervisor	The TMC / NSP / traffic_supervisor is responsible for the registration/deregistration and the monitoring of DG vehicles. In any case of an incident the TMC contacts the DG-vehicle directly. The TMC also offers a map display to further call centres and the fleet operator.
Call-centres	The call-centres are responsible to do their jobs in the case of an incident.
Fleet operator	The fleet operator is responsible for his fleet vehicles.

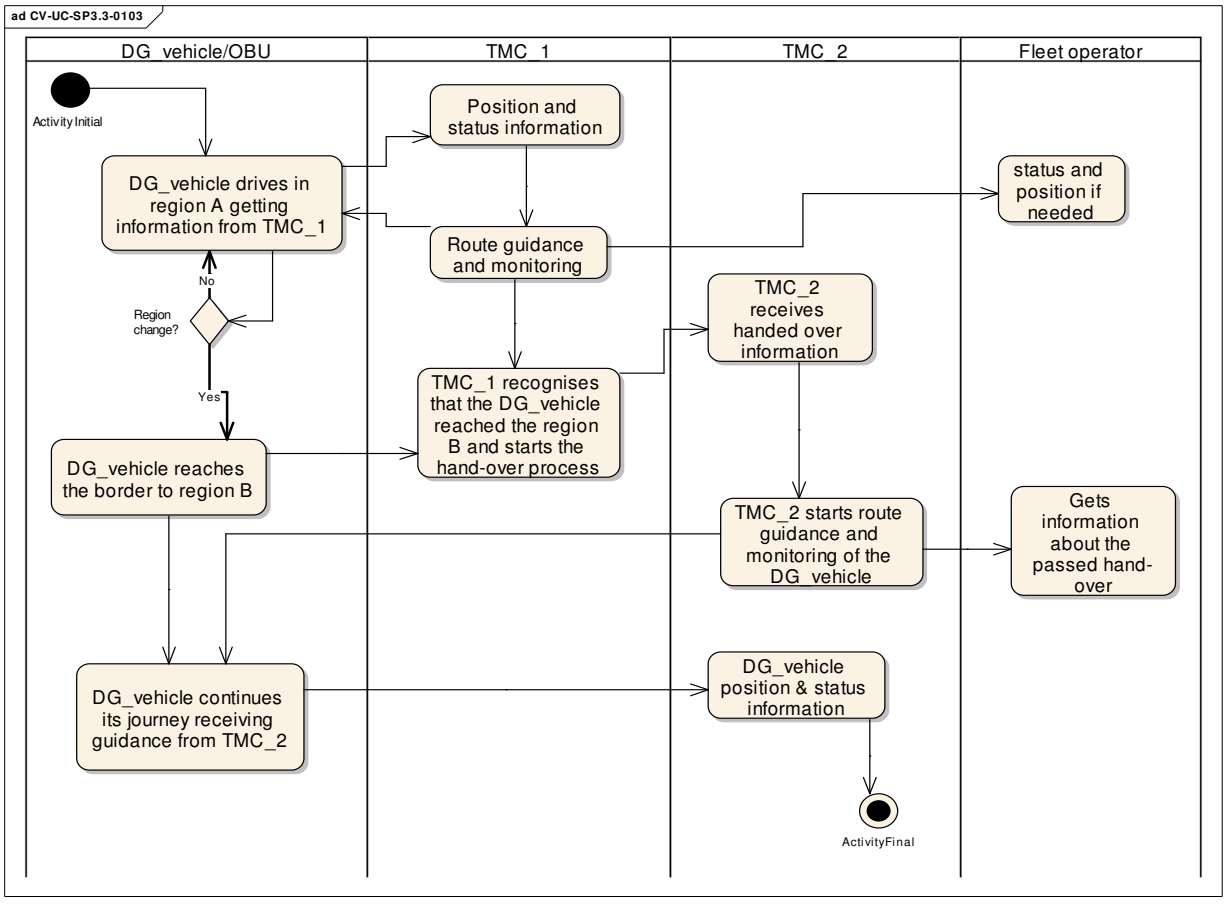

Figure 118: Activity diagram for DG vehicle hand-over

The table below describes the roles identified in the activity diagram and their responsibilities for this particular scenario.

Role	Responsibilities
DG_vehicle/OBU	The DG vehicle is responsible to send status & position information to the respective traffic management centre.
TMC_1	The traffic management centre 1 is responsible for the monitoring and the routing/guidance of a DG vehicle. It makes also the border-check and initialises the hand-over process.
TMC_2	The traffic management centre 2 makes the hand-over process and informs the fleet operator. Afterwards it is responsible for the monitoring and the route guidance of the DG vehicle.
Fleet operator	The fleet operator is responsible for its fleet vehicles.

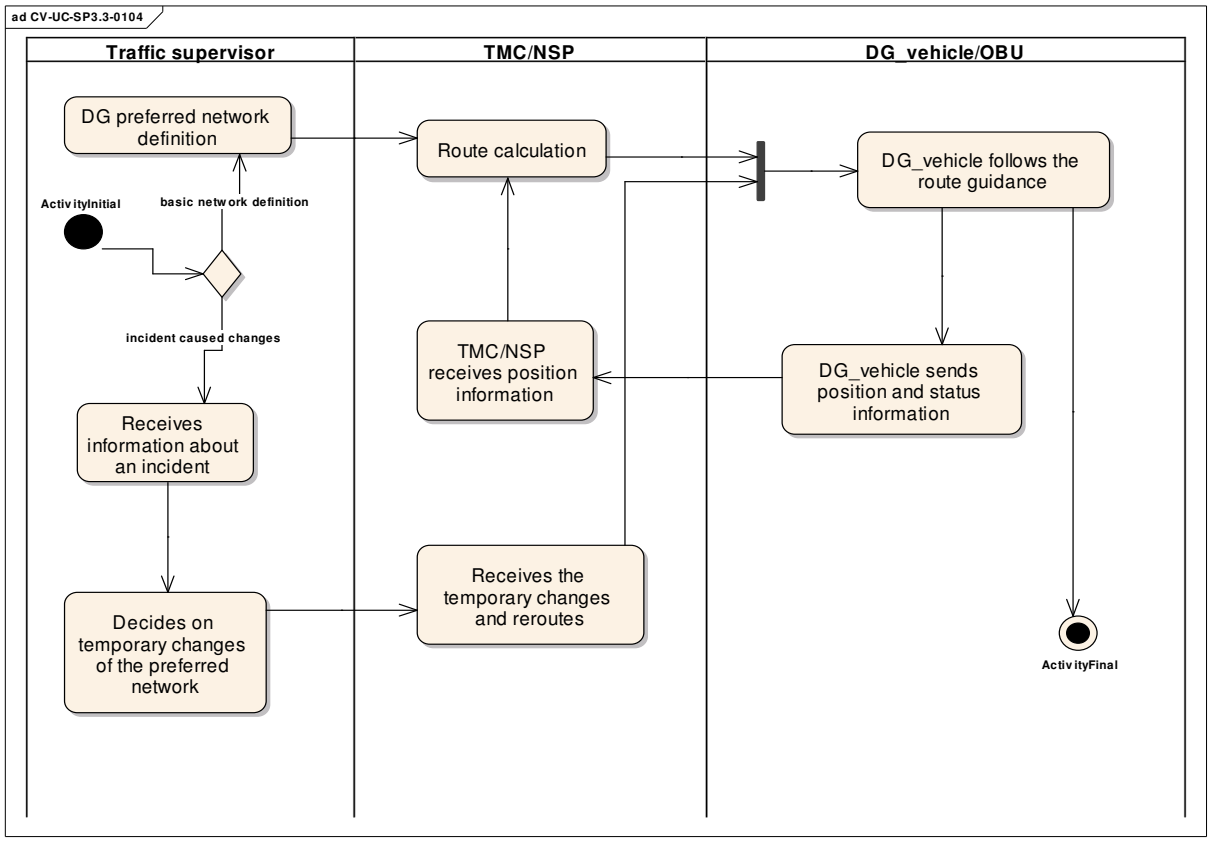

Figure 119: Activity diagram for DG preferred network management

The table below describes the roles identified in the activity diagram and their responsibilities for this particular scenario.

Role	Responsibilities
Traffic_supervisor	The traffic supervisor is responsible for the definition of the preferred network for DG vehicles. In case of an incident he decides on temporary changes of this defined network.
TMC/NSP	The traffic management centre is responsible for the route calculation and the routing. In case of temporary changes the traffic management centre makes the rerouting.
DG_vehicle/OBU	The DG vehicle follows the route guidance.

7.1.5 High level composite architecture

This chapter provides specifications of the high level composite architecture. The aim is to describe the overall architecture of the system and the partitioning into sub-systems and components.

The following diagram describes mainly the system components of the DG application within the CF&F sub-project. Furthermore the connections and interfaces towards middleware facilities are shown.

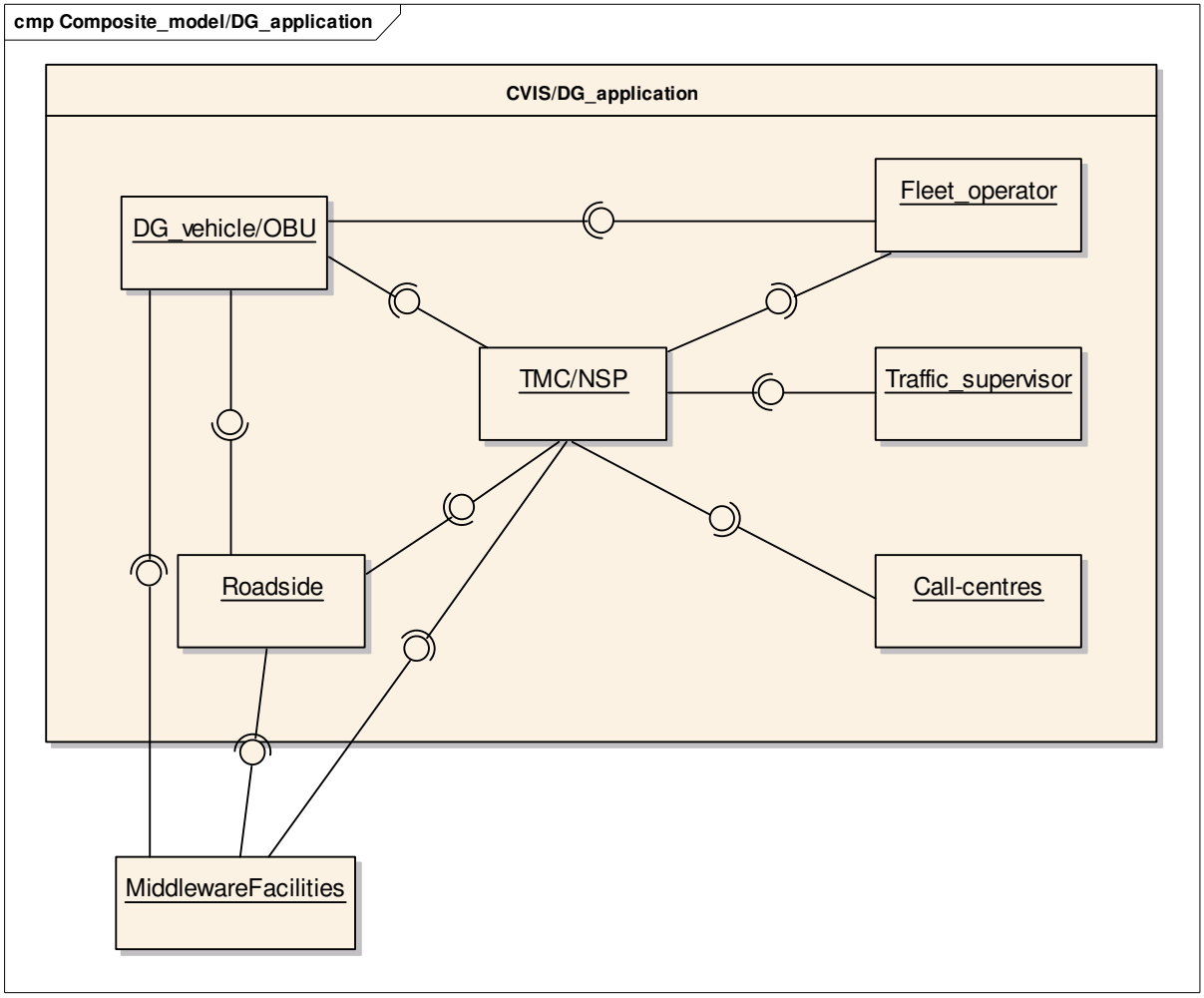

Figure 120: High level composite architecture for the DG application

7.1.6 Deployment model

The deployment diagram below describes the logical deployment structure of the DG Application.

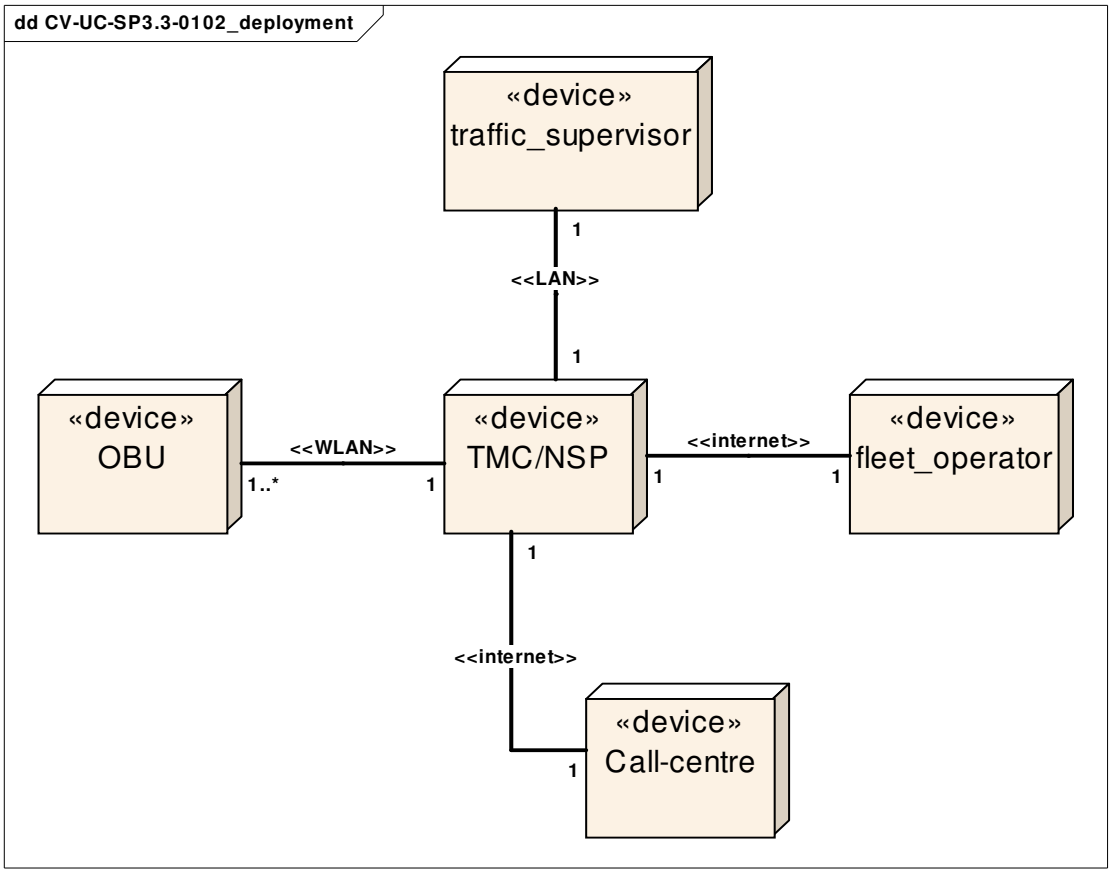

Figure 121: Deployment diagram for the DG vehicles monitoring

7.2 Parking zones

The parking zones application and its main services are introduced in this sub-section. Further details of the parking zones application can be found in the D.CFF.3.2 "Architecture Specification" document.

7.2.1 Overview

The parking zones application provides two main services, one for booking urban parking zones and one for booking highway resting areas. Using these services, a freight vehicle driver or a fleet operator can lookup highway resting areas and urban parking zones in a specific geographic area, for example along a route or at a destination. The operators, who operate the highway resting areas and urban parking zones, expose an interface, which the driver or fleet operator uses in the booking process.

This picture provides a non-technical overview of the parking zones application which is divided in the urban parking zones and the highway resting areas. Furthermore, the different actors like the CVIS equipped vehicle, the urban parking zone operator and the resting area

operator are illustrated.

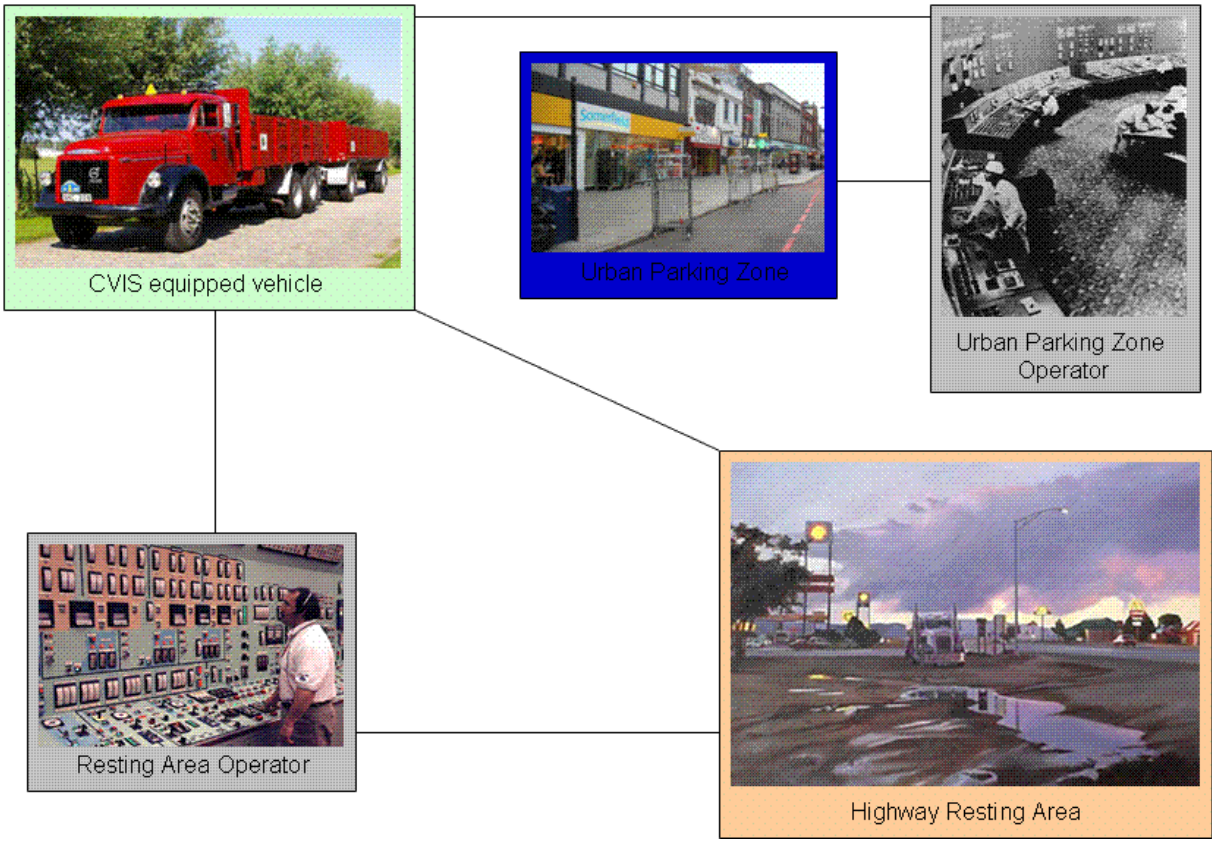

Figure 122: Overview of the parking zone application

Main use cases and system boundary

The parking zones application consists of use cases for booking urban parking zones and highway resting areas. The booking, in both cases, can be done by either a fleet operator or by the freight vehicle driver as illustrated in Figure 123.

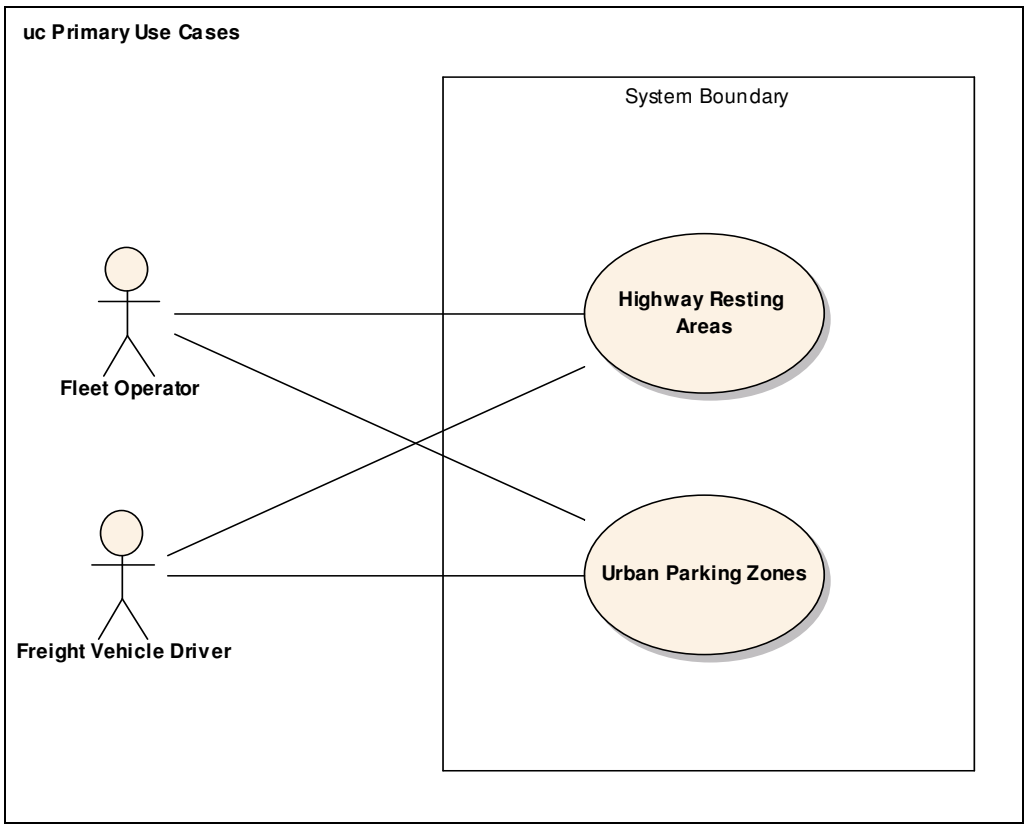

Figure 123: Use case model with system boundary for the parking zone use cases

7.2.2 Application programming interface

In the parking zones applications the driver and fleet operator are considered to be the external actors.

Figure 124 describes the interface as seen by these actors in the urban parking zone applications.

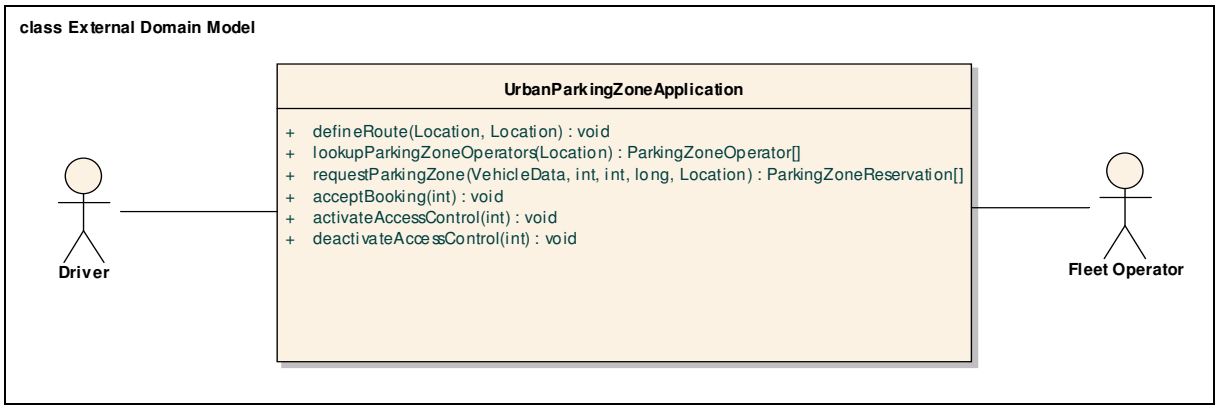

Figure 124: External interface of the urban parking zone application

Figure 125 exemplifies the usage of the urban parking zone services specifying interaction scenarios from the driver and the fleet operator respectively.

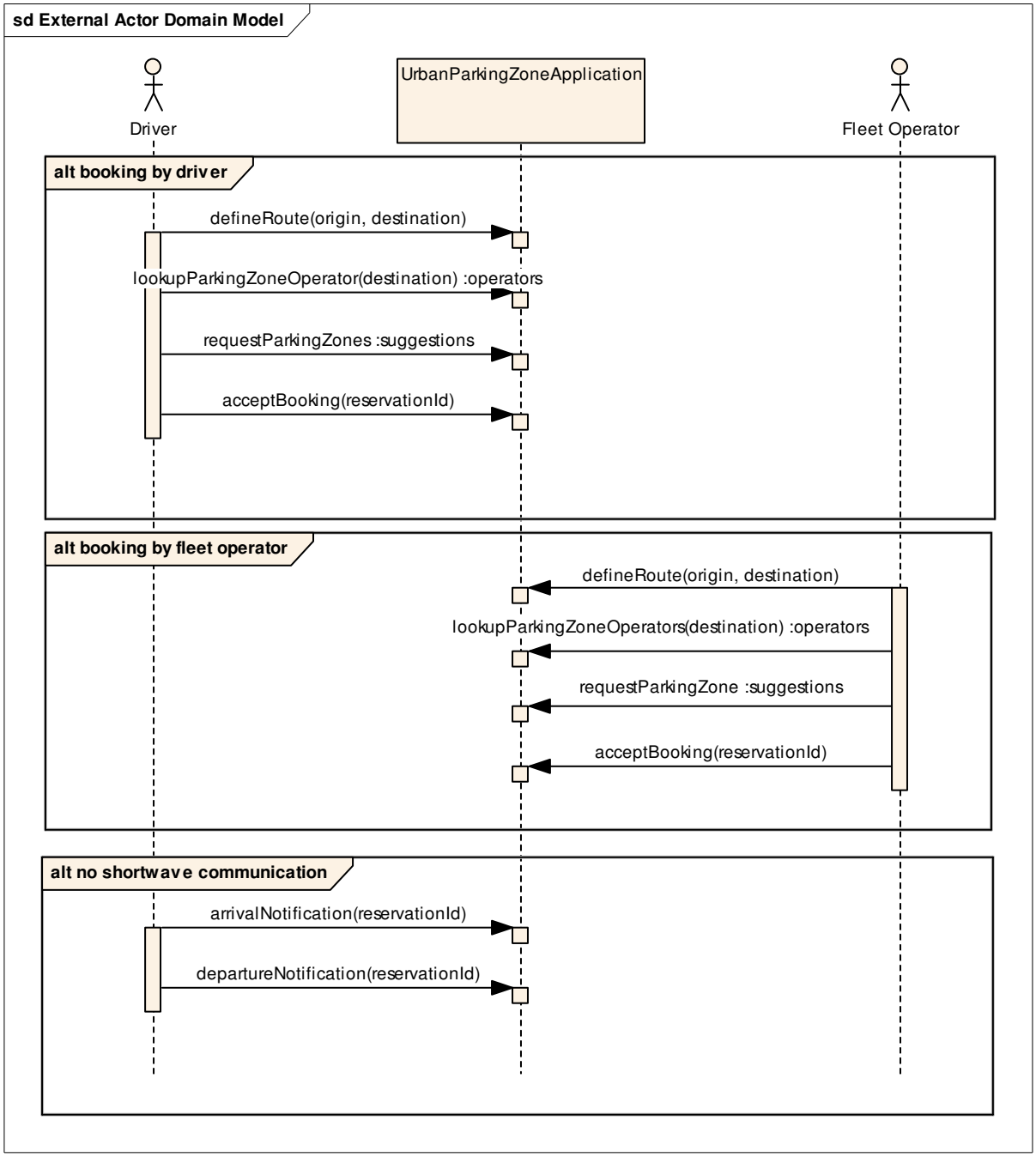

Figure 125: Sequence diagram describing external actor interaction with the urban parking zone application

Figure 126 describes the interface as seen by the fleet operator and the driver in the highway resting area applications.

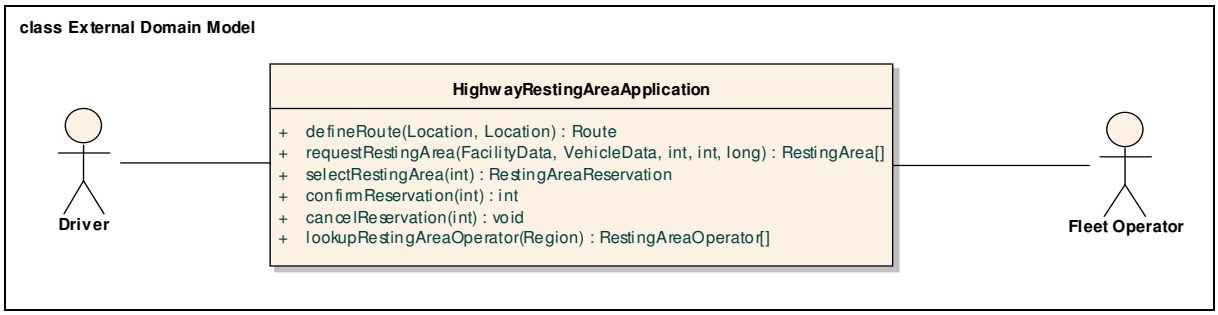

Figure 126: External interface of the highway resting area application

7.2.3 Information model

The figure below specifies the domain model for the parking zone application.

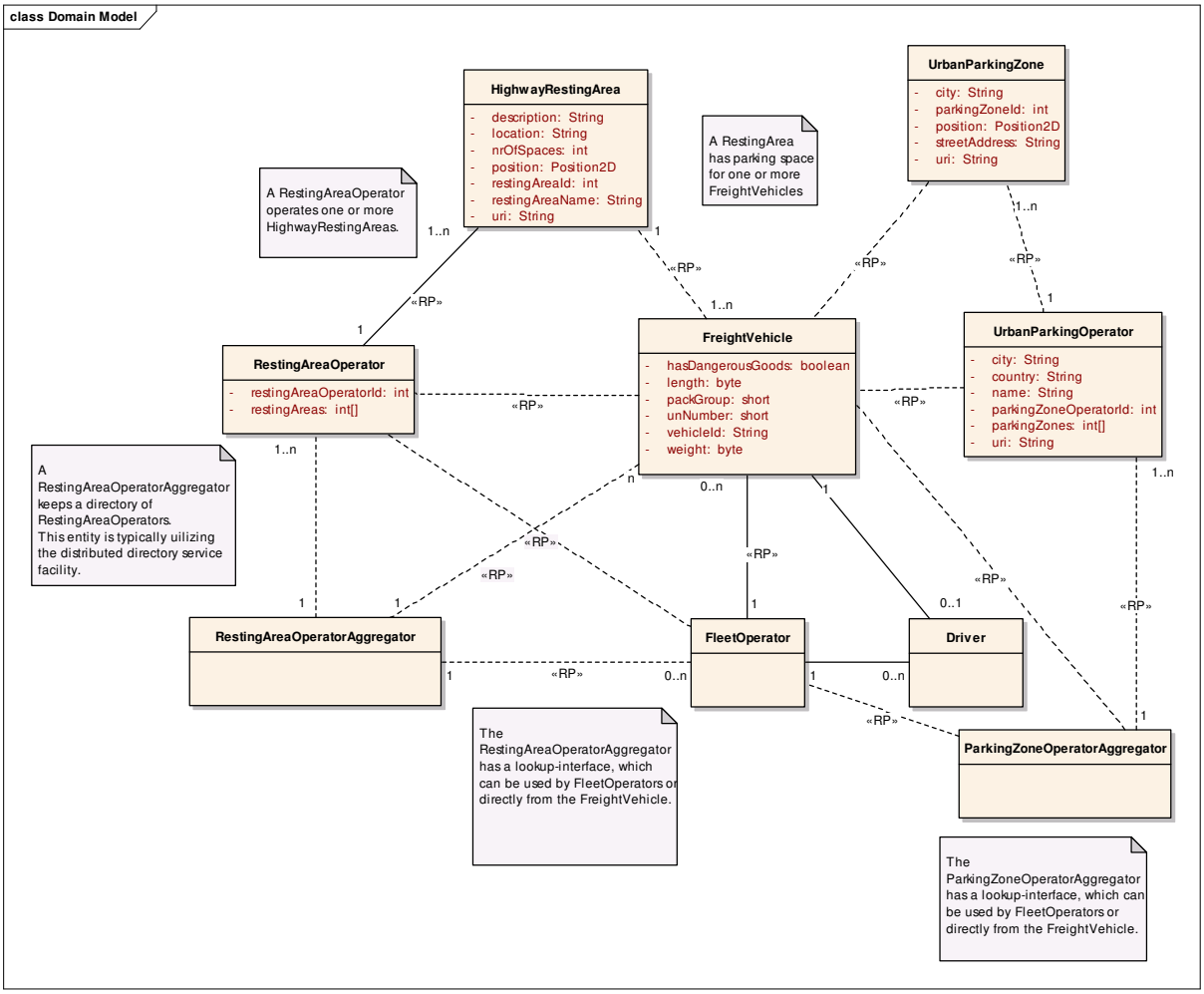

Figure 127: Domain model for the parking zone application

A brief description of the information entities is provided in the table below:

Entity Name	Entity Abbreviation	Entity Description
FleetOperator	FO	In the context of the CF&F parking zone application a fleet operator runs a transport management system, which is able to discover and connect to parking operators.
FreightVehicle	FV	A freight vehicle contains a client system (CVIS host platform) able to connect to the outside world using a CALM router.
Driver	D	The driver interacts with the in-vehicle CVIS host system.
RestingAreaOperator	RO	The resting area operator is a parking operator who manages highway resting areas.
UrbanParkingOperator	UPO	An urban parking operator is a parking operator who manages urban parking zones.
HighwayRestingArea	HRA	A highway resting area is managed by and booked via a parking provider. The resting area contains a CVIS road-side unit, which enable connections to its parking operator and the freight vehicle as well as access control.
UrbanParkingZone	Z	An urban parking zone is an area in an urban environment for loading-unloading of freight vehicle. An urban parking zone is managed by and booked via a parking provider.
RestingAreaOperatorAggreagor	ROA	A resting area operator aggregator keeps a registry of resting area operators. In the parking zones applications this role can be played by a navigation provider.
ParkingZoneOperatorAggregator	POA	A parking zone operator aggregator keeps a registry of urban parking zone operators. In the parking zones applications this role can be played by a navigation provider.

7.2.4 Interaction model

The interactions of the two main use cases of the parking zones application:

Urban parking zone,

DG vehicle route guidance,

are specified below using UML activity diagrams.

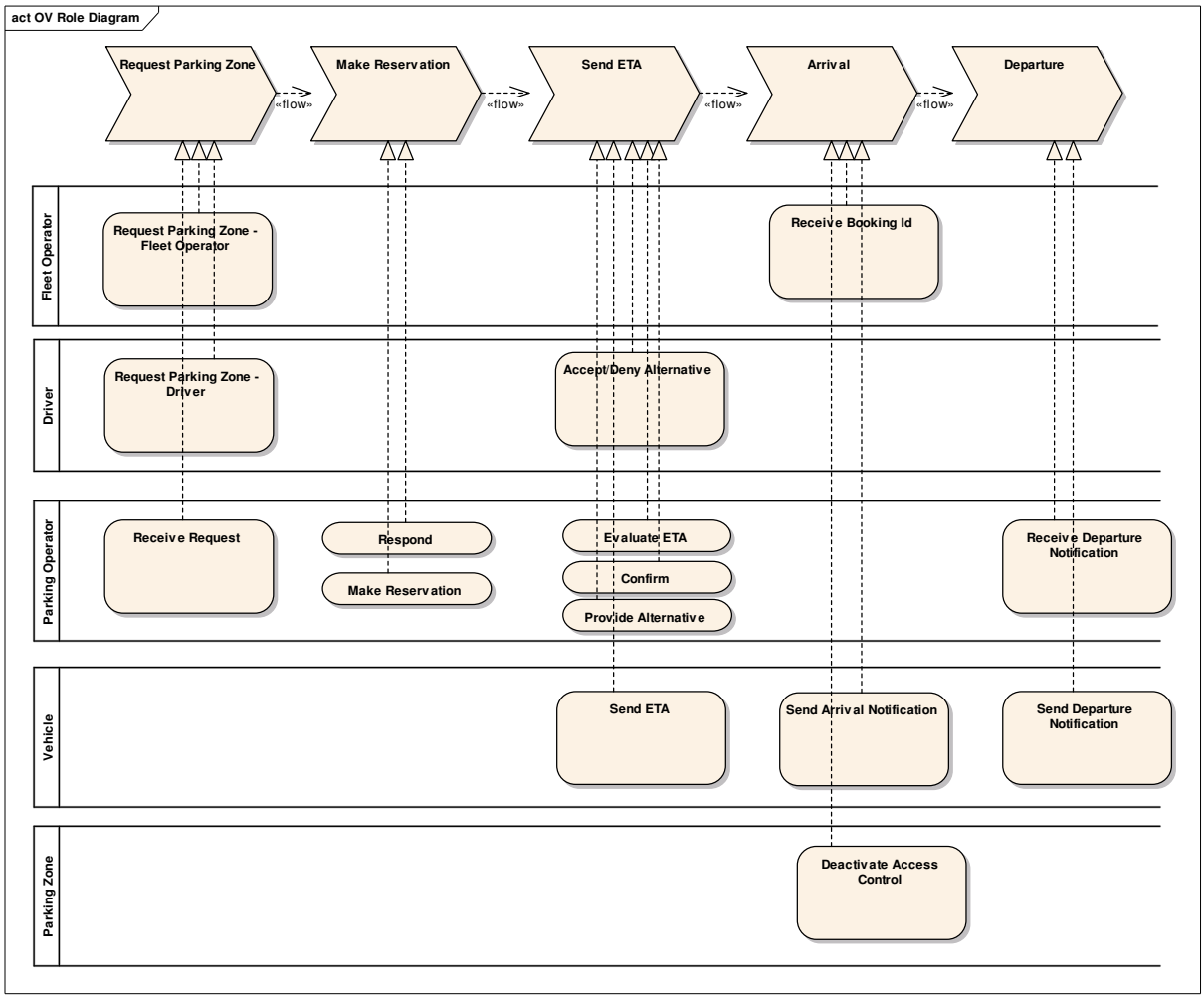

Figure 128 Role diagram for urban parking zone application

The table below describes the roles identified in the activity diagram and their responsibilities for this particular scenario.

Role	Responsibilities
Fleet operator	The fleet operator or the driver is responsible for requesting an urban parking zone.
Driver	The fleet operator or the driver is responsible for requesting an urban parking zone. The driver informs the parking operator about time of arrival and duration for the request.

Role	Responsibilities
Parking Operator	The parking operator is responsible for making the reservation, keeping its parking database updated.
Vehicle	The vehicle provides ETA to the parking operator and receives and provides the reservation ID number.
Parking zone	The parking zone may have access control facilities, which are controlled by the parking operator.

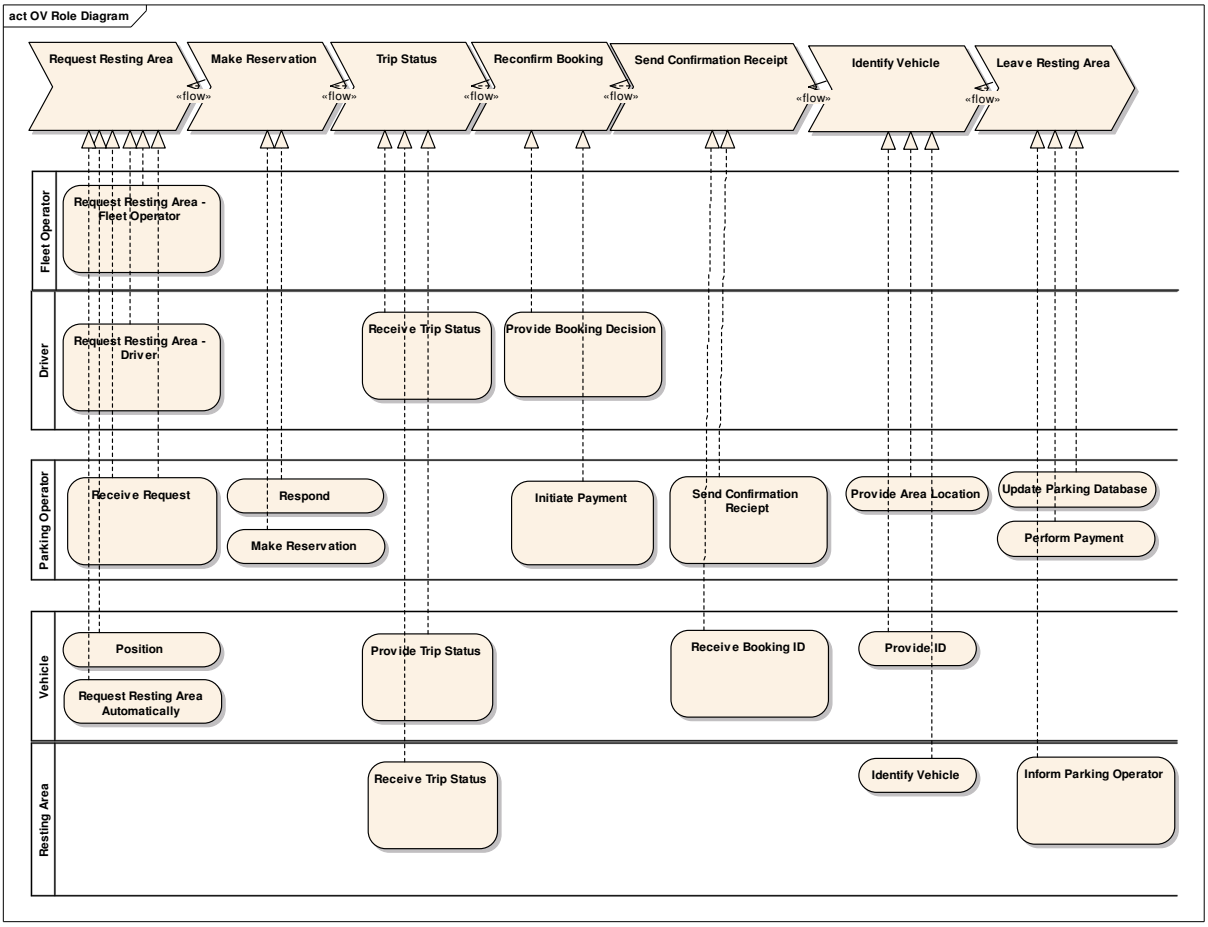

Figure 129: Diagram for highway resting area application

The table below describes the roles identified in the activity diagram and their responsibilities for this particular scenario.

Role	Responsibilities
Fleet operator	The fleet operator or the driver is responsible for requesting a resting area.
Driver	The fleet operator or the driver is responsible for requesting a resting area. The driver informs the parking operator about time of arrival and duration for the request.

Role	Responsibilities
Parking Operator	The parking operator is responsible for making the reservation, manage payment, keeping its parking database updated.
Vehicle	The vehicle provides its position to the request maker provides status of the trip to the driver and the resting area and receives and provides the reservation ID number.
Resting Area	The resting area is responsible for identifying and authorizing the vehicle.

7.2.5 High level composite architecture

The composite diagram for the parking zones application shows the in vehicle service component in relation with the other system components especially actors of the parking zones application.

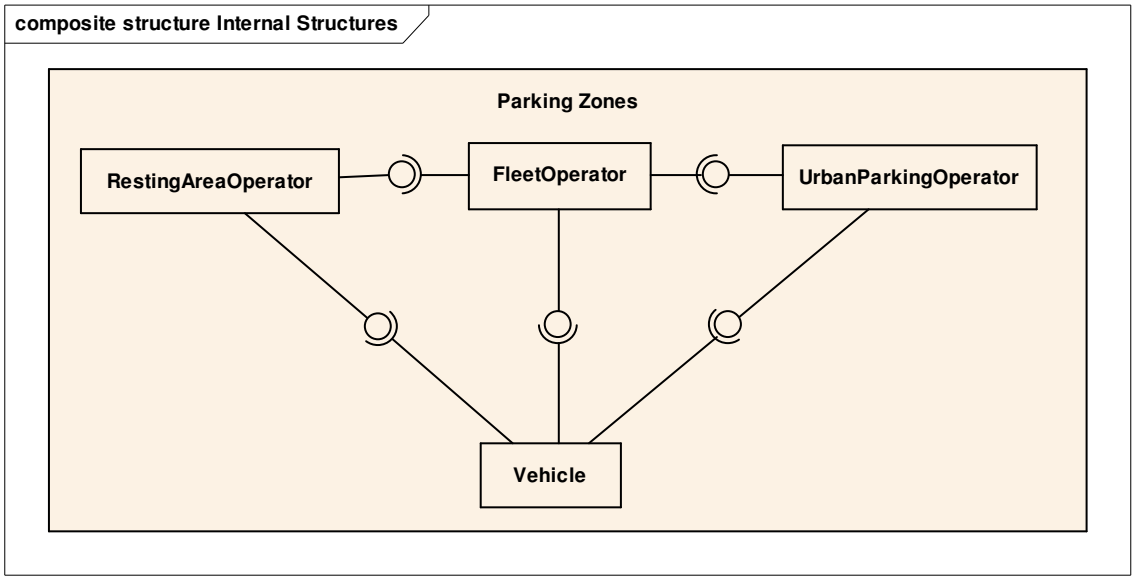

Figure 130: High level composite architecture for the parking zone service

7.2.6 Deployment model

The deployment model describes the logical deployment of the services of the parking zone application onto the CVIS infrastructure.

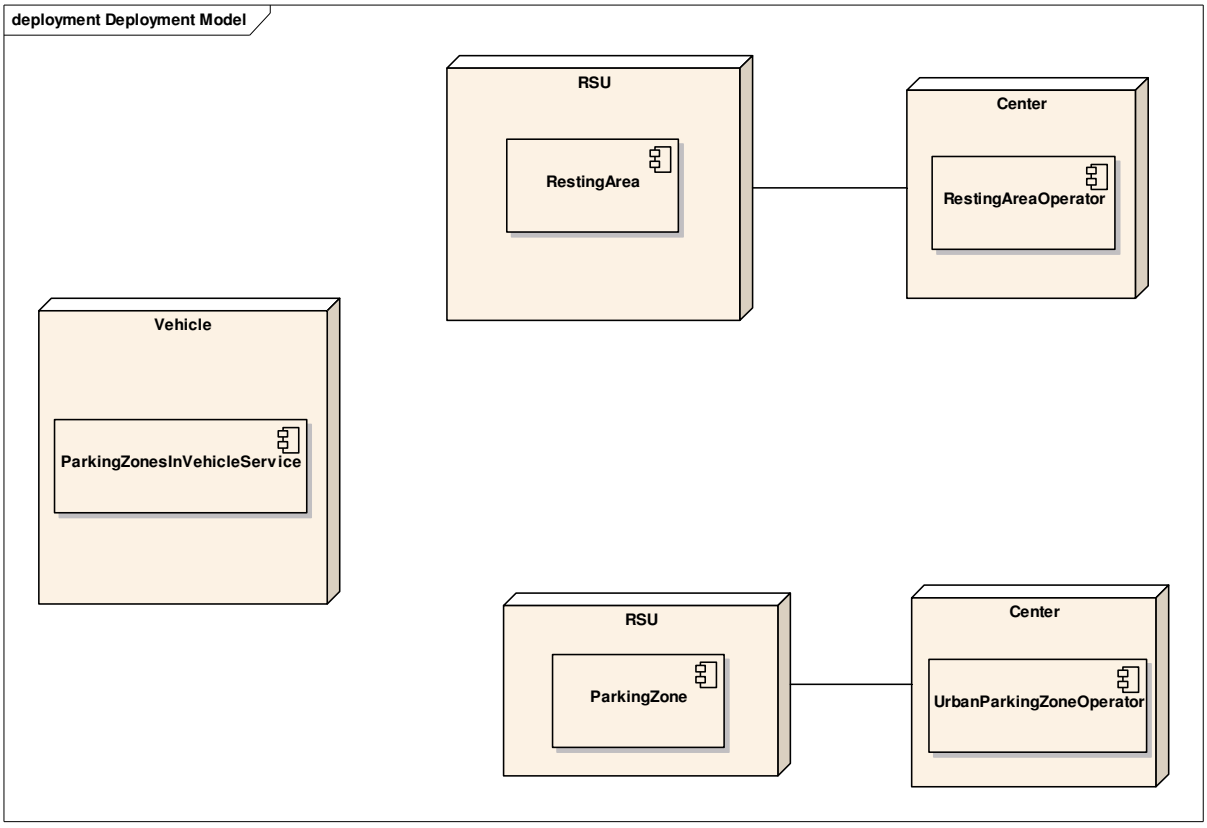

Figure 131: Deployment diagram of the parking zone applications

7.3 Access control

The access control application and its main services are introduced in this sub-section. Further details of the access control application can be found in the D.CFF.3.2 "Architecture Specification" document.

7.3.1 Overview

The basic ideas of the access control application is to monitor vehicles approaching sensitive zones in order to allow/deny the access, as a preventive safety measure to avoid accidents and as a tool to control dynamically traffic conditions in restricted areas. This could be achieved by means of an "always-on" seamless communication between incoming vehicles and the infrastructure. This is depicted in Figure 132.

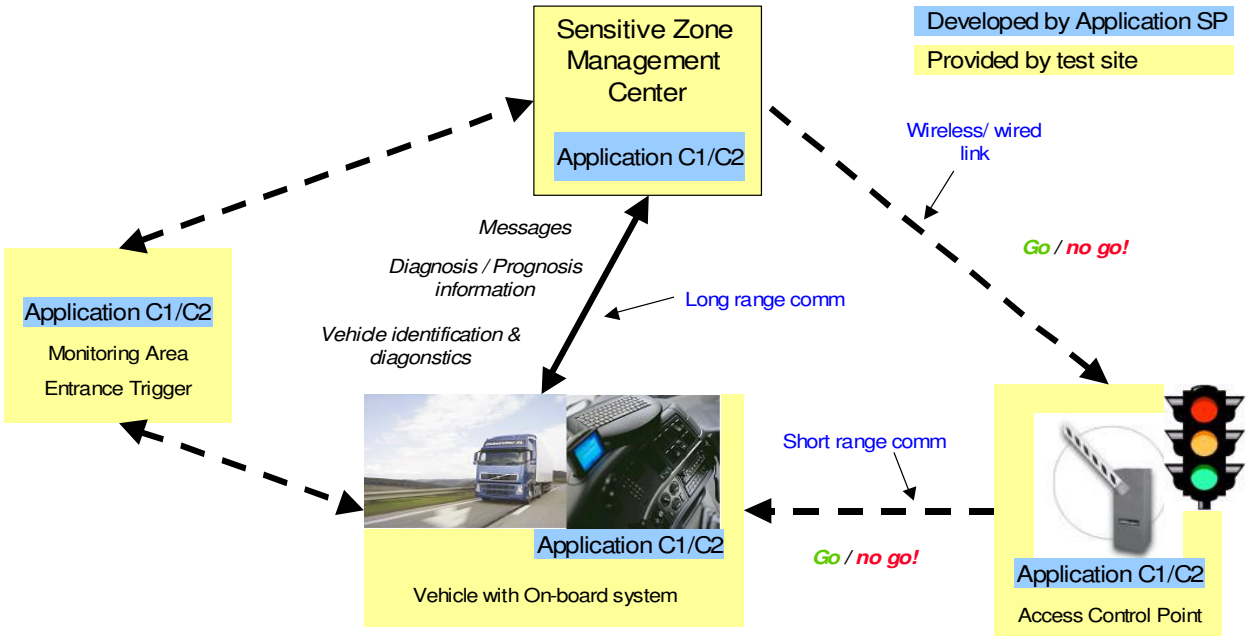

Figure 132: Access control

Involved CVIS technologies are dynamic geo-fencing policies and simultaneous wireless data transaction over two different bearers: medium range V2I & I2V and long range bi-directional "Vehicle to Control Centre" (over 2G-3G network).

After detecting the entrance inside a predefined monitoring area a remote monitoring session is activated and the vehicle is tracked while approaching the critical access area. The approaching vehicle automatically transmits self-diagnostics and driving status data to the access control centre.

Based on predefined policies and real time potential risk assessment, the access control centre provides the approaching vehicle with preventive grant or denial to access the critical area.

Information, logs and feedback on the decision taken must be available to be sent to the fleet manager and to the driver.

Two uses cases have been defined:

Approaching access control area,

Decision making and information feedback.

These are specified in the use case model below.

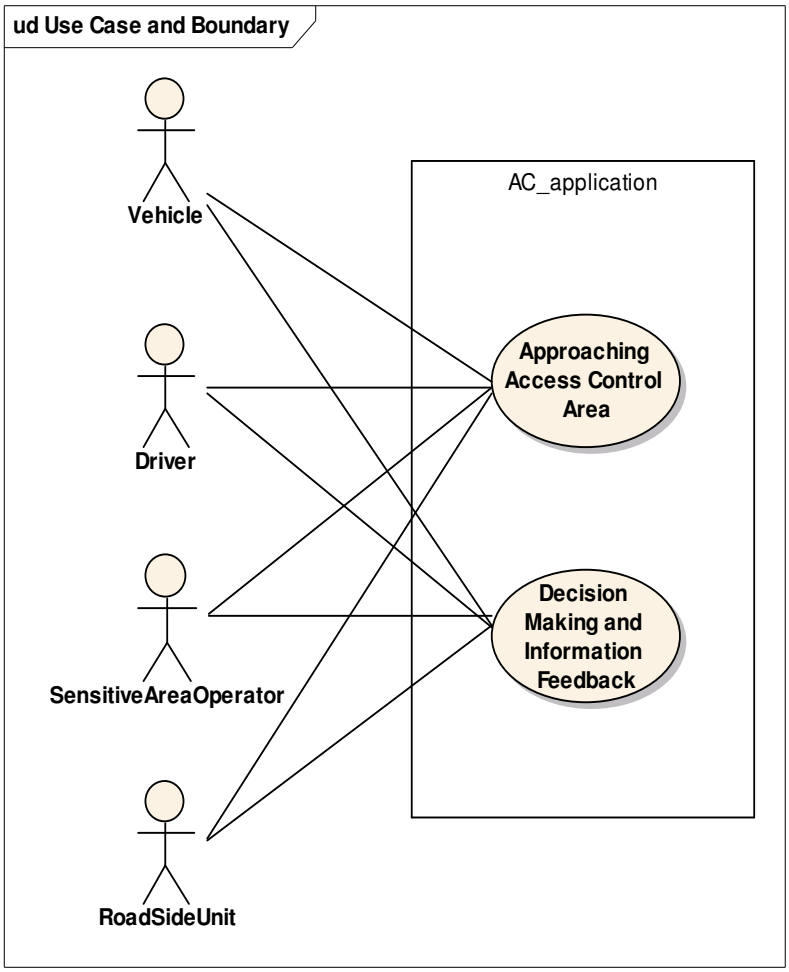

Figure 133: Main use cases and system boundary for the access control application

The use cases are:

Approaching access control area; In this use case the vehicle equipped with the CVIS platform, transmits vehicle diagnostics data and driving behaviour to the access control manager while approaching the sensitive area.

Decision making and information feedback; In this use case the vehicle equipped with the CVIS platform, which interacts with the access control manager to obtain access granted or denied to the predefined area. Information feedback on the decision taken will be sent to the fleet manager (and or other interested parties) and to the driver.

The main actors using the system are:

Driver.

Vehicle.

RoadSideUnit.

SensitiveAreaOperator.

7.3.2 Application programming interface

Interface specifications are an essential part of the architecture and system specification as the interactions are performed through the interfaces. The cooperation and relationships between the different parts of one system are enabled by these interfaces. The following diagram gives an overview of the interfaces recognised in connection with the access control application.

In the access control applications the "Driver", "Access Control operator" and "Fleet Manager operator" are considered to be the external actors.

The figure below describes the interface as seen by these actors in the access control applications.

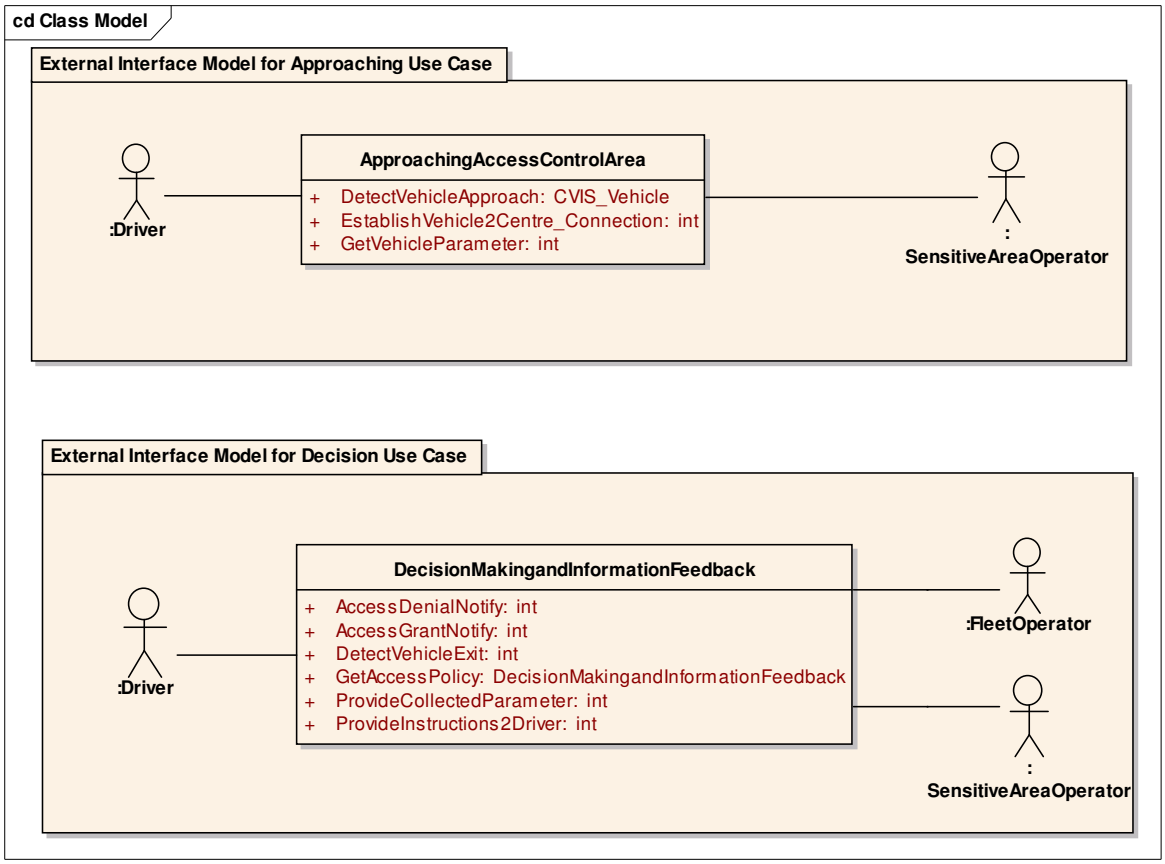

Figure 134: External interface of the access control application

7.3.3 Information model

The figure below specifies the domain model for the access control application.

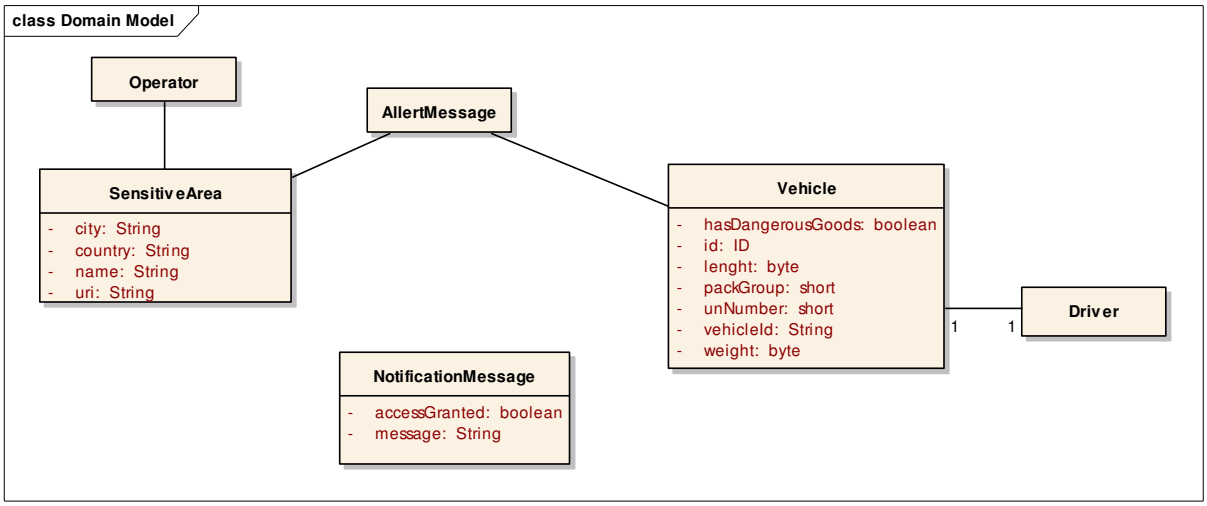

Figure 135: Access control information model

7.3.4 Interaction model

The interactions of the two main use cases of the access control application:

Approaching access control area,

Decision making and information feedback,

are specified below using UML activity diagrams.

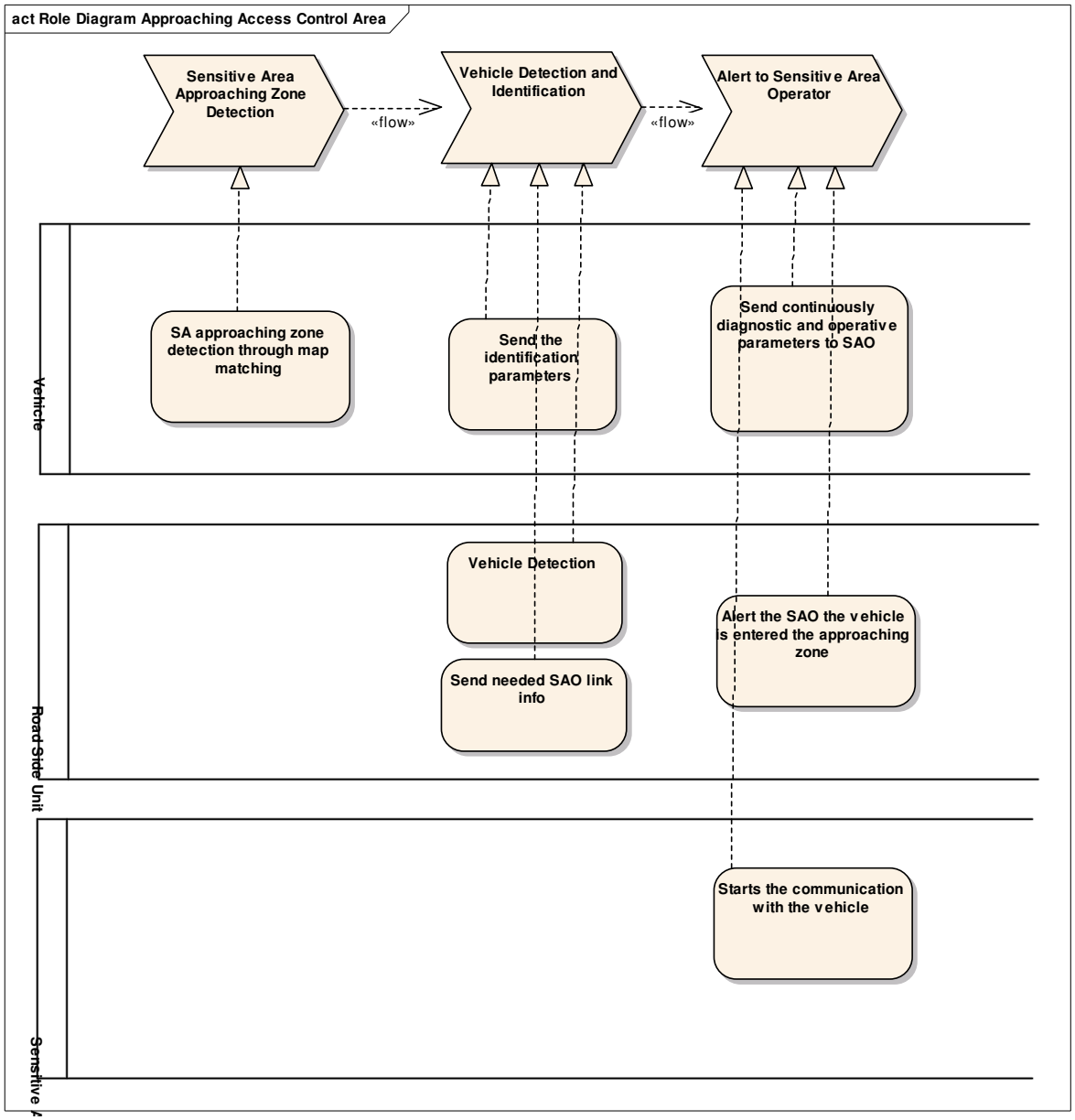

Figure 136: Role diagram for approaching access control area

The table below describes the roles identified in the activity diagram and their responsibilities for this particular scenario.

Role	Responsibilities
Vehicle	The vehicle is responsible for identifying its entrance in the monitoring area by means of map matching technology.
Sensitive area operator	When alerted about vehicle entrance in the monitoring area, he manages the monitoring process.
Road-side unit	The road-side unit is responsible for identifying its entrance in the monitoring of the vehicle by means of geo-fencing technology and transmitting the related connection parameters to the vehicle.

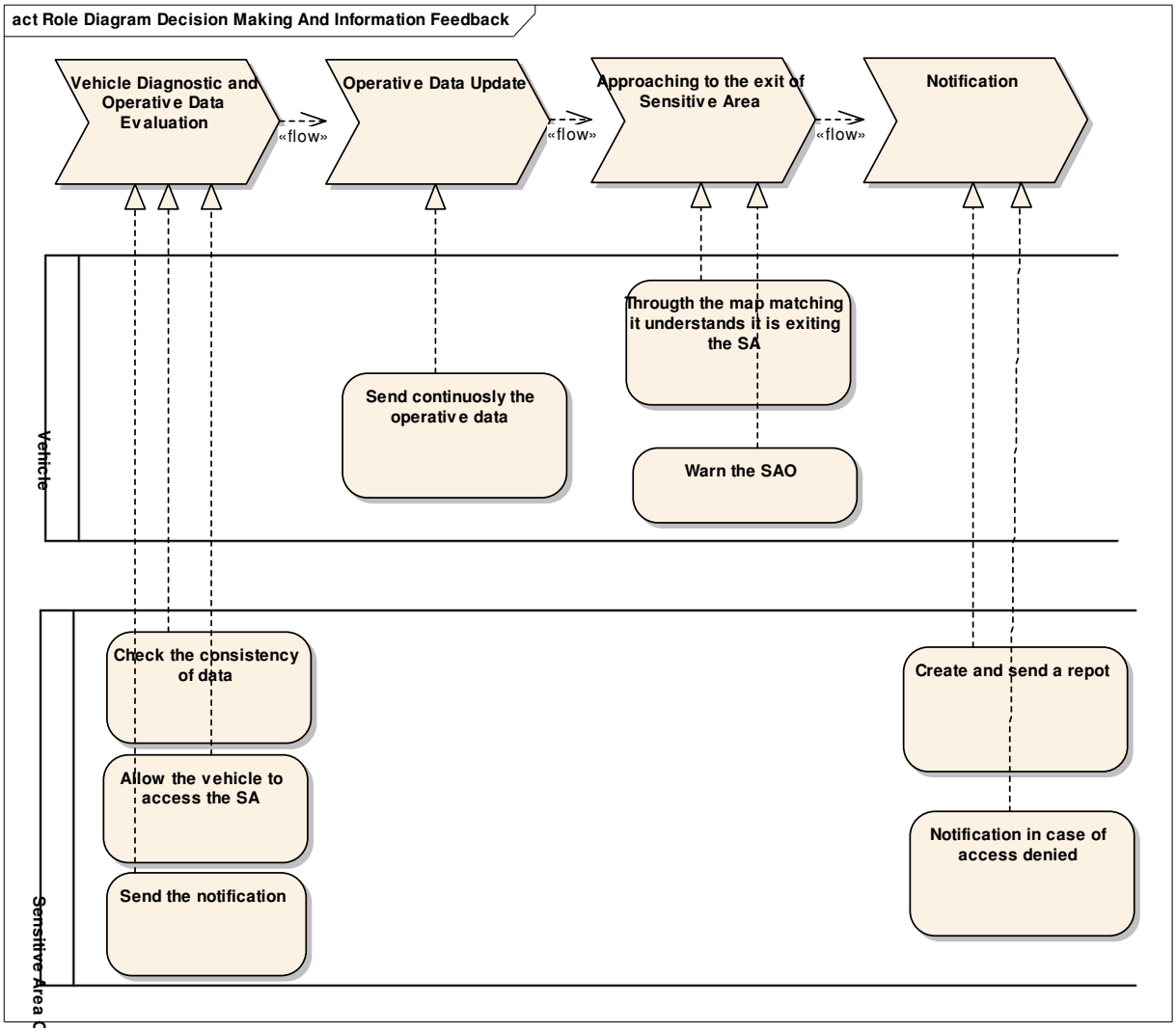

Figure 137: Role diagram for decision making and information feedback

The table below describes the roles identified in the activity diagram and their responsibilities for this particular scenario.

Role	Responsibilities
Vehicle	The vehicle is responsible for formatting the on-board parameters to the access control centre by using the proper policy.
Sensitive Area Operator	The sensitive area operator is in charge of collecting the parameters from the monitored vehicles, processing them according to the applicable policy and managing the access to the area.

7.3.5 High level composite architecture

The access control composite diagram specifies the main system components of the access control application.

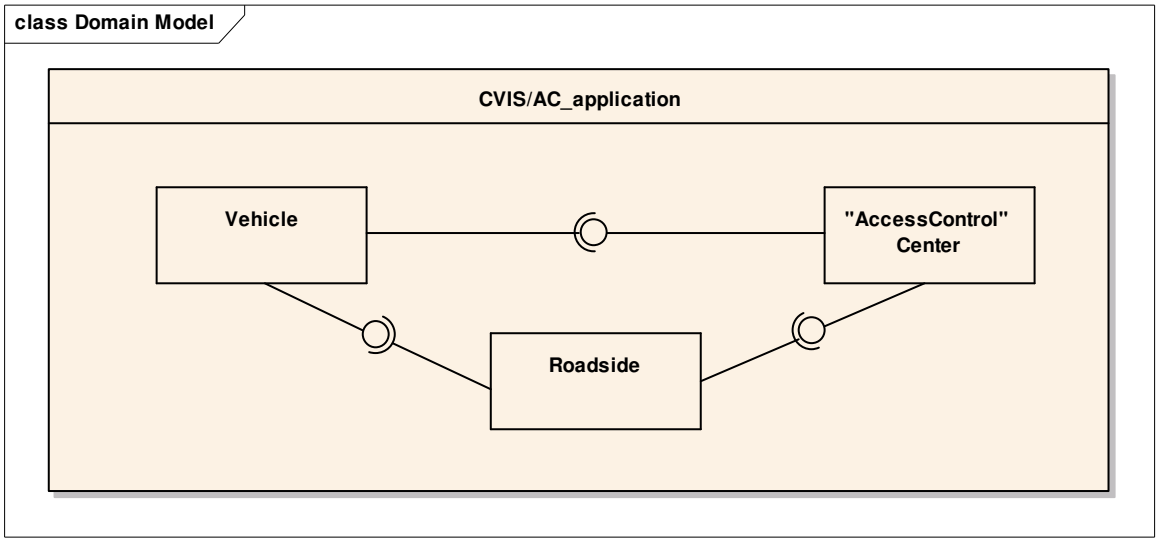

Figure 138: High level composite architecture for the access control application

7.3.6 Deployment model

The deployment models describe the logical deployment onto the CVIS infrastructure, that is, the deployment of the above specified components to physical nodes.

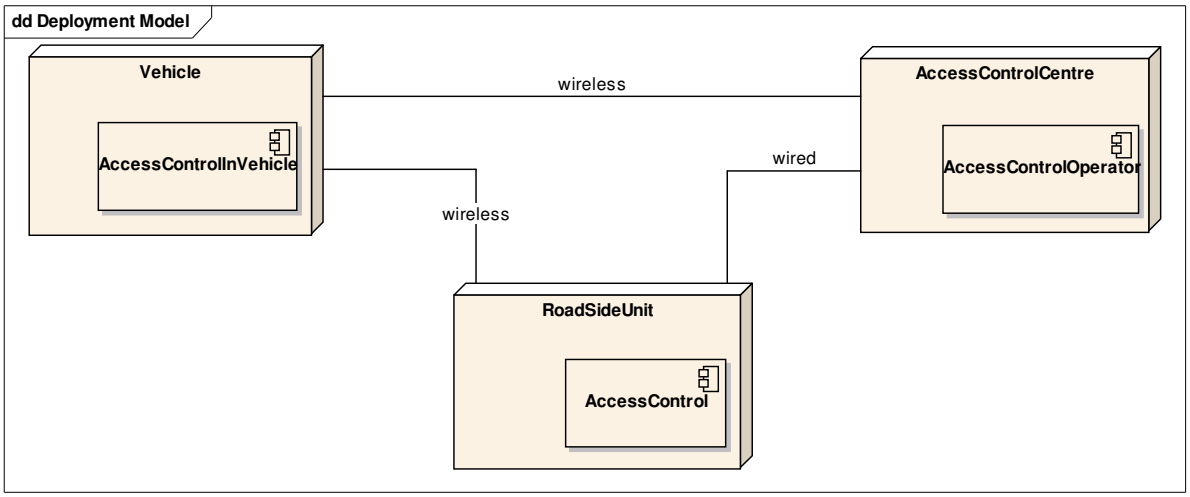

Figure 139: Deployment diagram of the access control applications

All wired communication can be replaced with wireless communication. Wireless communication can not be replaced by wired communication.

7.4 Cooperative traveller assistance

The "Cooperative Traveller Assistance" (CTA) application and its main services are introduced in this sub-section. Further details of the CTA application can be found in the D.CINT.3.2 "Architecture Specification" document.

7.4.1 Overview

The CTA application consists of three main services / sub applications that will provide assistance to travellers and drivers of other vehicles, e.g. heavy goods vehicles, but not public transport and emergency service vehicles. The services provided by each of the CTA sub applications are as follows:

Pre-trip and on-trip planning: Drivers can plan their trips across the inter-urban road network according to their need to travel, their specific origin and destination within the Inter-urban road network, plus the current and forecast traffic conditions. In addition drivers can change their previously prepared trip plans, or produce plans for the first time, whilst their journeys are in progress.

On-trip seamless service with tracking and rerouting if needed: the service centre takes care of drivers' requests providing information and (re)routing guidance depending on individual driver preferences and vehicle characteristics.

Vehicle data feeding to traffic control centres: the collection of vehicle and planning data enhances the determination of current and forecast traffic conditions so that they can be combined and used in the preparation of trip plans. This data can also be used to calculate strategies to assist with the management of the traffic using the Inter-urban road

network.

The UML use case model with the system boundary for CTA is shown in Figure 140.

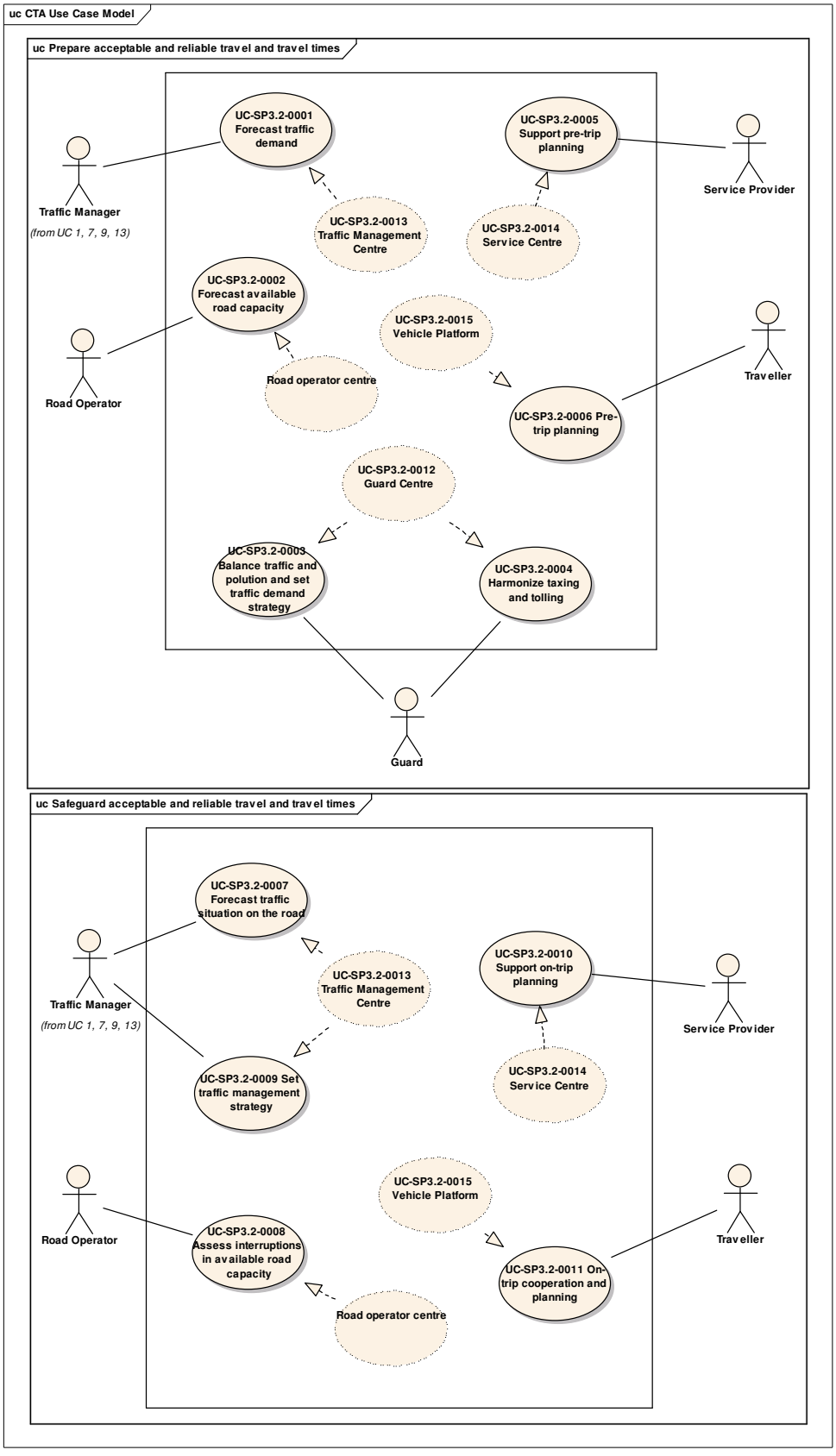

Figure 140: CTA use case model with system boundary

The use cases are as follows (for elaborated descriptions of the use cases see D. CINT D2.1):

Forecast traffic demand: Forecast the traffic demand on the inter-urban road network.

Forecast road capacity: Forecast the available road capacity on the inter-urban road network given the planned road construction works and the forecasted weather conditions

Set traffic demand strategy: Given the forecasted availability of road capacity and the forecasted traffic demand, set the boundary conditions for the acceptable traffic demand on the interurban road network.

Harmonising taxing and tolling: Harmonising taxing and tolling to allow travellers to plan their trip taking taxing and tolling into account

Support pre-trip planning: To support travellers in planning their trip on beforehand.

Pre-trip planning: To support travellers in planning their trip on beforehand.

Forecast traffic situation on the road: Forecast the traffic situation on the interurban road network

Assess interruptions in available road capacity: Forecast the interruptions in the available capacity in the interurban road network, due to e.g. weather conditions, congestion, bridge operations, smaller road works, et cetera.

Set traffic management strategy: Forecast the interruptions in the available capacity in the interurban road network, due to e.g. weather conditions, congestion, bridge operations, smaller road works, et cetera.

Support on-trip planning: Support travellers, e.g. vehicle drivers with their on-trip planning

On-trip cooperation (planning): Support travellers, e.g. vehicle drivers with their on-trip planning.

Support guard by guard service centre: To support the guard in building up an image of the forecasted quality of mobility and environment and to set a traffic demand strategy.

Support traffic manager by traffic management centre: To support the traffic manager in building up an image of the forecasted traffic demand on the interurban road network and of an image of the currently evolving traffic situation on the interurban road network.

Support service provider by service centre: To support the service provider in supporting travellers, e.g. vehicle drivers in their on-trip planning.

Support traveller, e.g. vehicle driver by vehicle: To support the traveller, e.g. vehicle driver in their on-trip planning.

The actors and their needs and responsibilities are described in the following:

"Traffic manager: This is a human entity that manages the operation of the CINT (and other) applications forming the traffic management system located in the "Traffic Management Centre" (TMC) that is responsible for the inter-urban road network. The traffic manager is able to manage how the applications in the TMC operate and the information that is made available to vehicle drivers and travellers. Additionally the traffic manager can decide on the way that vehicles are able to use the network, e.g. OPEN/CLOSE lanes, and set speed limits, etc.

Traveller: This human entity represents the vehicle driver when that entity is not driving its vehicle. Its main purpose is to enable the vehicle driver to carry out pre-trip planning from outside the vehicle. For this purpose the traveller will interface with the CINT applications using a mobile system. This may be a PDA or a static computer such as a PC. If the traveller chooses to do their trip planning whilst in the vehicle, then will become the vehicle driver human entity - see definition below.

Road operator: This role is responsible for the condition and usage of the interurban road network. It will manage the way that the inter-urban road network operates and the provision of information to the CINT applications in the traffic management

Service provider: The service provider is a physical entity or an organisation that controls a series of physical entities that can run applications forming part of the CINT system. These applications are capable of providing "services" to end users. For CINT these "services" will provide information directly to the end users about relevant aspects of the dynamic traffic situation, current speed and other regulations. There will also be "services" for trip planning and other similar facilities. There may be one service provider from which all these types of information are available, or several providers from which one or a range of types of information are available. Any of the types of information may be provided to all end users either on a commercial (subscription) basis through a service centre (see above), or free of charge.

Guard: **(Guard for mobility and QoE)**: - this entity represents the government departments and organisations that are responsible for safe guarding the quality of mobility and the environment in a country / region / municipality. Typically this entity will be concerned with the way in which mobility is provided and the quality of the environment over the longer term (5, 10, 15 years).

7.4.2 Application programming interface

The methods exposed at the vehicle and service centre side respectively are specified in Figure 141.

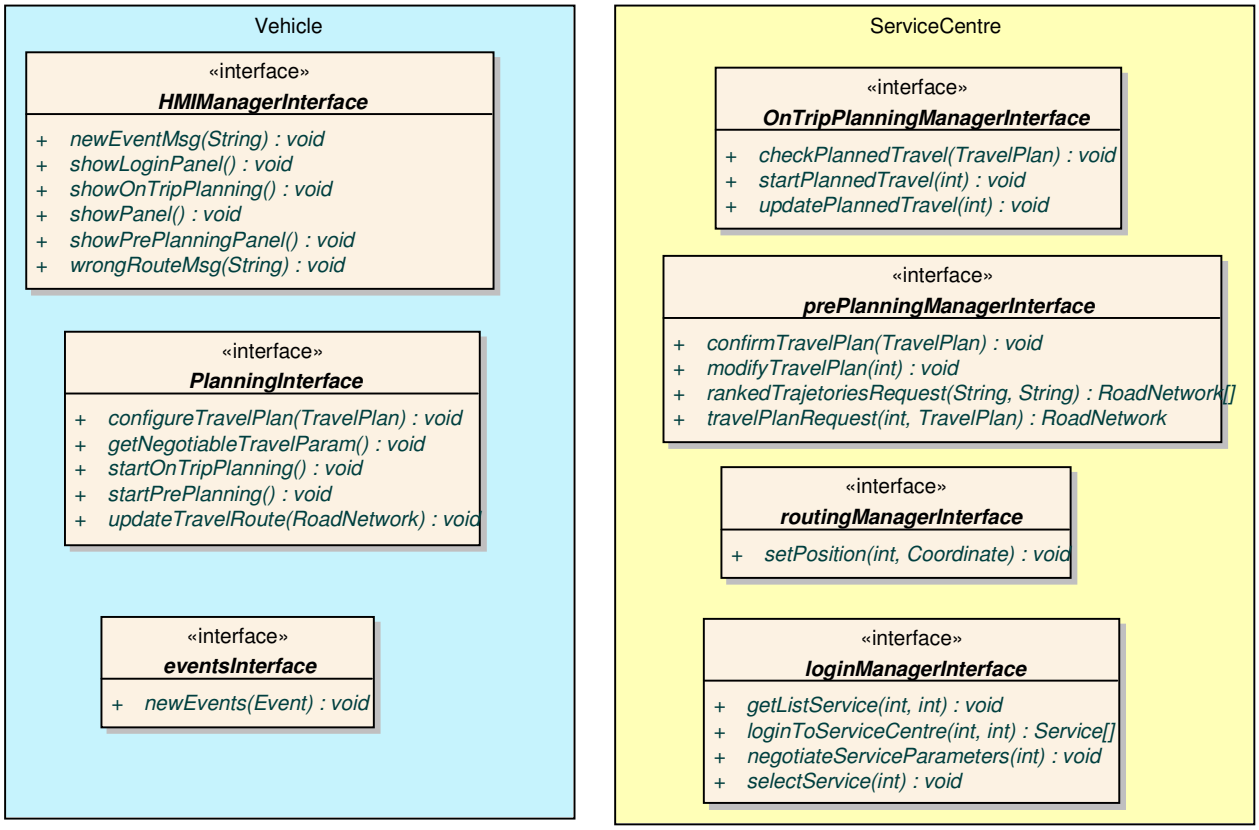

Figure 141: CTA API

The provided methods are elaborated in the following tables.

HMIManagerInterface (vehicle)

Method	Type	Notes
newEventMsg (*String*)	public: *void*	Show a new event
showLoginPanel ()	public: *void*	Show login panel
showPanel ()	public: *void*	Show generic panel
showPrePlanningPanel ()	public: *void*	Show PrePlanning Panel
wrongRouteMsg (*String*)	public: *void*	Warning at the driver because out of route
showOnTripPlanningPanel ()	public: *void*	Show the OnTripPlanning Panel

PreOnTripPlanningInterface (vehicle)

Method	Type	Notes
configureTravelPlan (*TravelPlan*)	public: *void*	This method allows setting the travel parameters. TravelPlan is a public class to be defined.
startPrePlanning ()	public: *void*	This method starts the PrePlanning component.
startOnTripPlanning ()	public: *void*	This method starts the OnTripPlanning component.
getNegotiableTravelParam ()	public: void	This method gets the available negotiable travel parameters.
updateTravelRoute (RoadNetwork)	public: void	This method allows updating a planned travel.

eventsInterface (vehicle)

Method	Type	Notes
newEvents (*Event*)	public: *void*	This method receives new events sent by the eventsManager component on the service centre

prePlanningManagerInterface (service centre)

Method	Type	Notes
travelPlanRequest (*int, TravelPlan*)	public: *RoadNetwork*	This method manages the travel plan request coming from vehicles.
rankedTrajetoriesRequest (*String, String*)	public: *RoadNetwork[]*	This method manages the ranked trajectories request coming from vehicles
modifyTravelPlan (*int*)	public: *void*	This method allows the modification of planned travel.
confirmTravelPlan (*TravelPlan*)	public: *void*	This method manages the confirmation of a travel plan.

OnTripPlanningManagerInterface (service centre)

Method	Type	Notes
startPlannedTravel (*int*)	public: *void*	This method manages new vehicles that start planned travel.
checkPlannedTravel (*TravelPlan*)	public: *void*	This method allows the checking of planned travel.
updatePlannedTravel (*int*)	public: *void*	This method allows the modification and update planned travel

routingManagerInterface (service centre)

Method	Type	Notes
setPosition (*int, Coordinate*)	public: *void*	This method saves the actual position of each vehicles connected to the service centre.

7.4.3 Information model

The following figure shows the high level domain information model for the "Co-operative Traveller Assistance" (CTA) application.

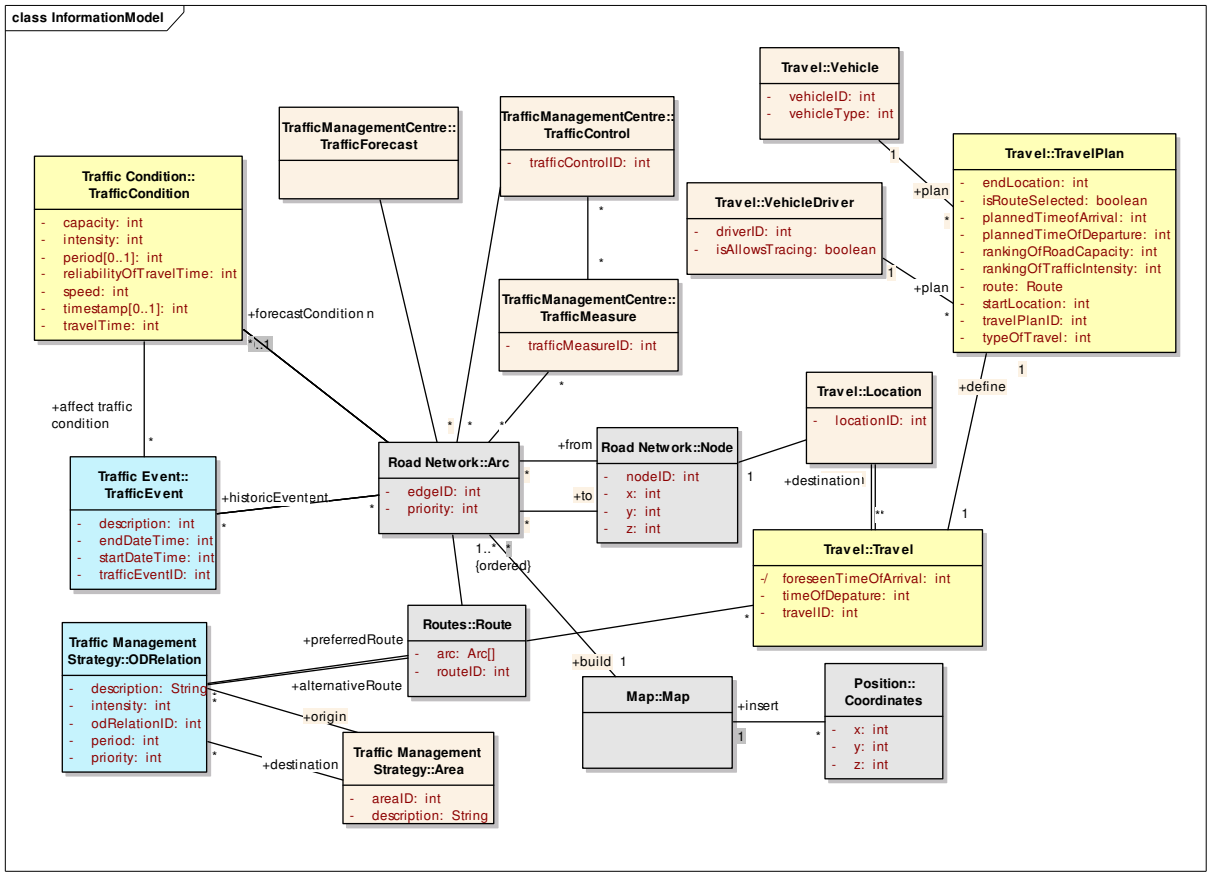

Figure 142: Domain information model for CTA

D.CINT.3.2 provides detailed descriptions of these concepts.

7.4.4 Interaction model

This section provides the main interaction processes of the CTA application at a high level. More elaborated descriptions are provided in D.CINT.3.2.

The activity diagram in Figure 143 provides the overall description of the interaction process carried out to fulfil the *pre-trip planning* and the *support on-trip planning* use cases.

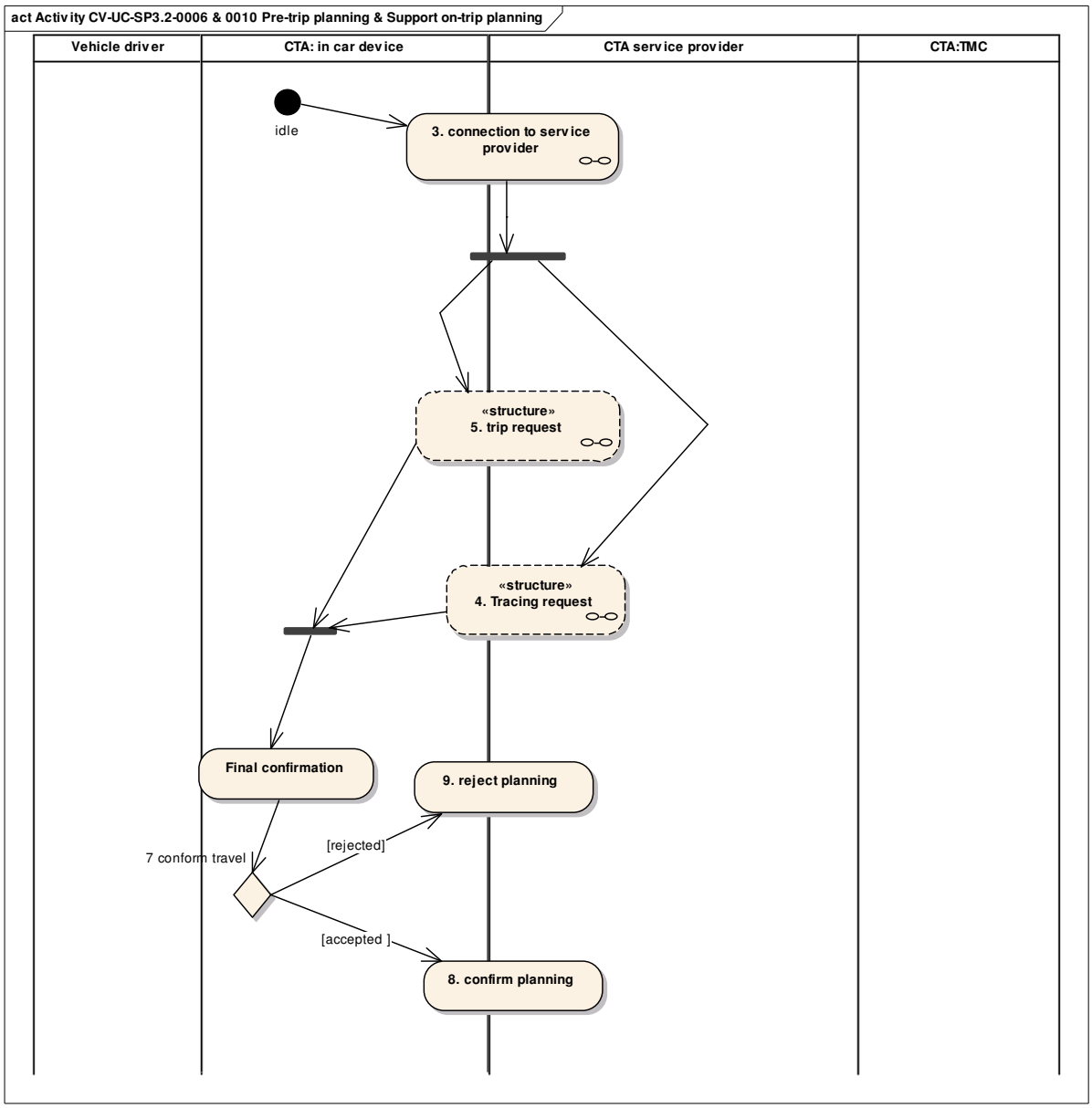

Figure 143: Activity diagram for pre-trip planning and support for on-trip planning

The activity diagram in Figure 144 provides the overall description of the interaction process carried out to fulfil the *on-trip cooperation (planning)* and *the support traveller, c.q. vehicle driver by vehicle* use cases.

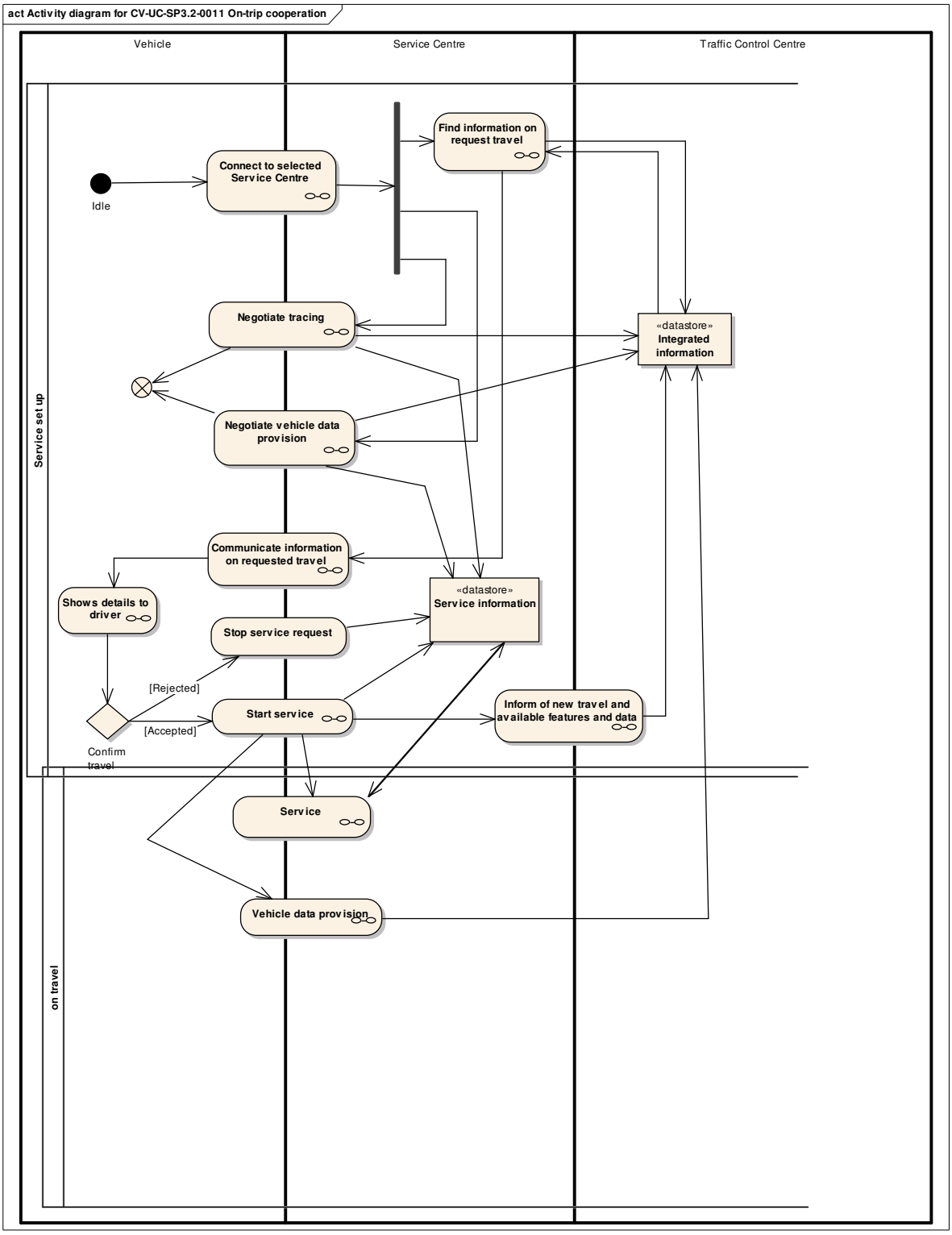

Figure 144: Activity diagram for on-trip co-operation

This activity diagram depicts two main phases:

Initial planning (service set up);

Service while on travel.

In the initial planning a driver may:

Load a previous defined travel plan;

Update a previous defined travel plan;

Create travel plan from scratch.

Despite of the previous cases the travel plan is always updated:

By confirmation of the travel;

By setting the real departure time;

By sending the updated traffic conditions.

The driver starts selecting a service centre in a list. How the list is created and maintained is out of the CVIS project scope. There may actually be more than one service centre and the driver may have to make a choice between two or more competing centres

The activity diagram in Figure 145 provides the overall description of the interaction process carried out to fulfil the *support pre-trip planning* and the *support service provider by service centre* use cases.

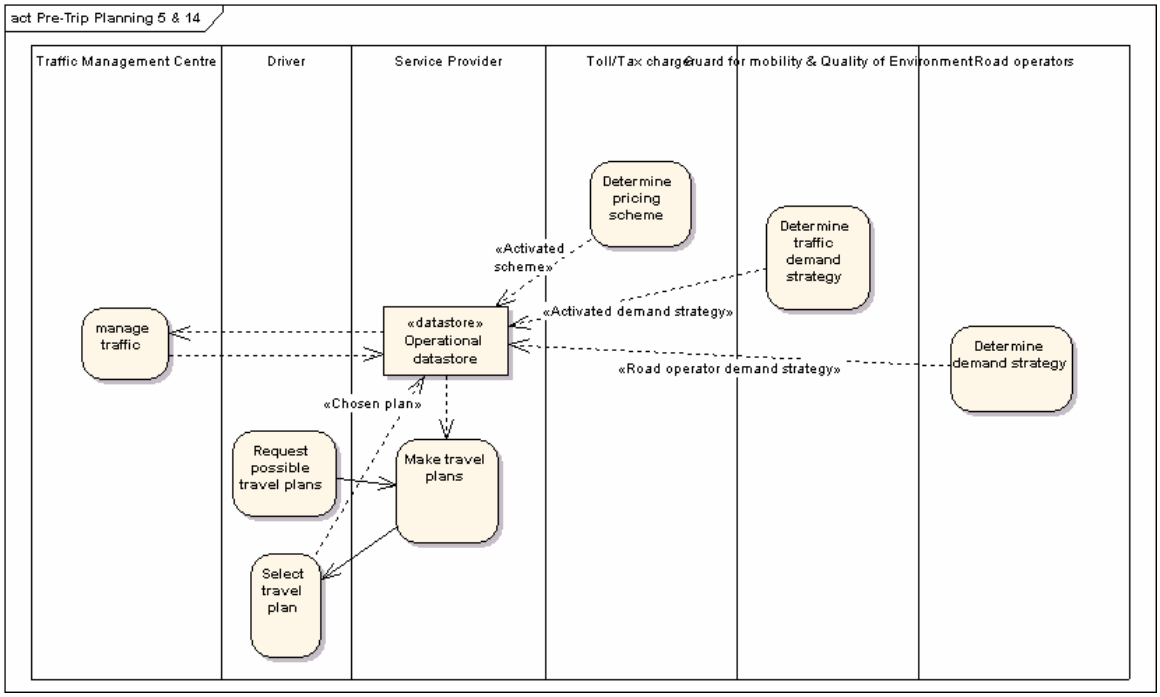

Figure 145: Activity diagram for trip planning and service support

The activity diagram in Figure 146 provides the overall description of the interaction process carried out to fulfil the *harmonising taxing and tolling* use case.

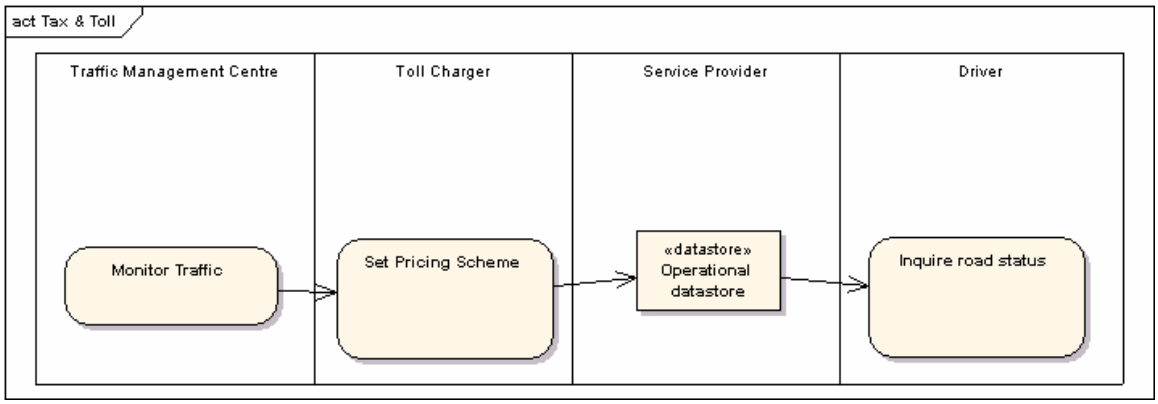

Figure 146: Activity diagram for tax and toll harmonisation

7.4.5 High level composite architecture

The high-level composite architecture for the "Co-operative Traveller Assistance" (CTA) application is shown in Figure 147.

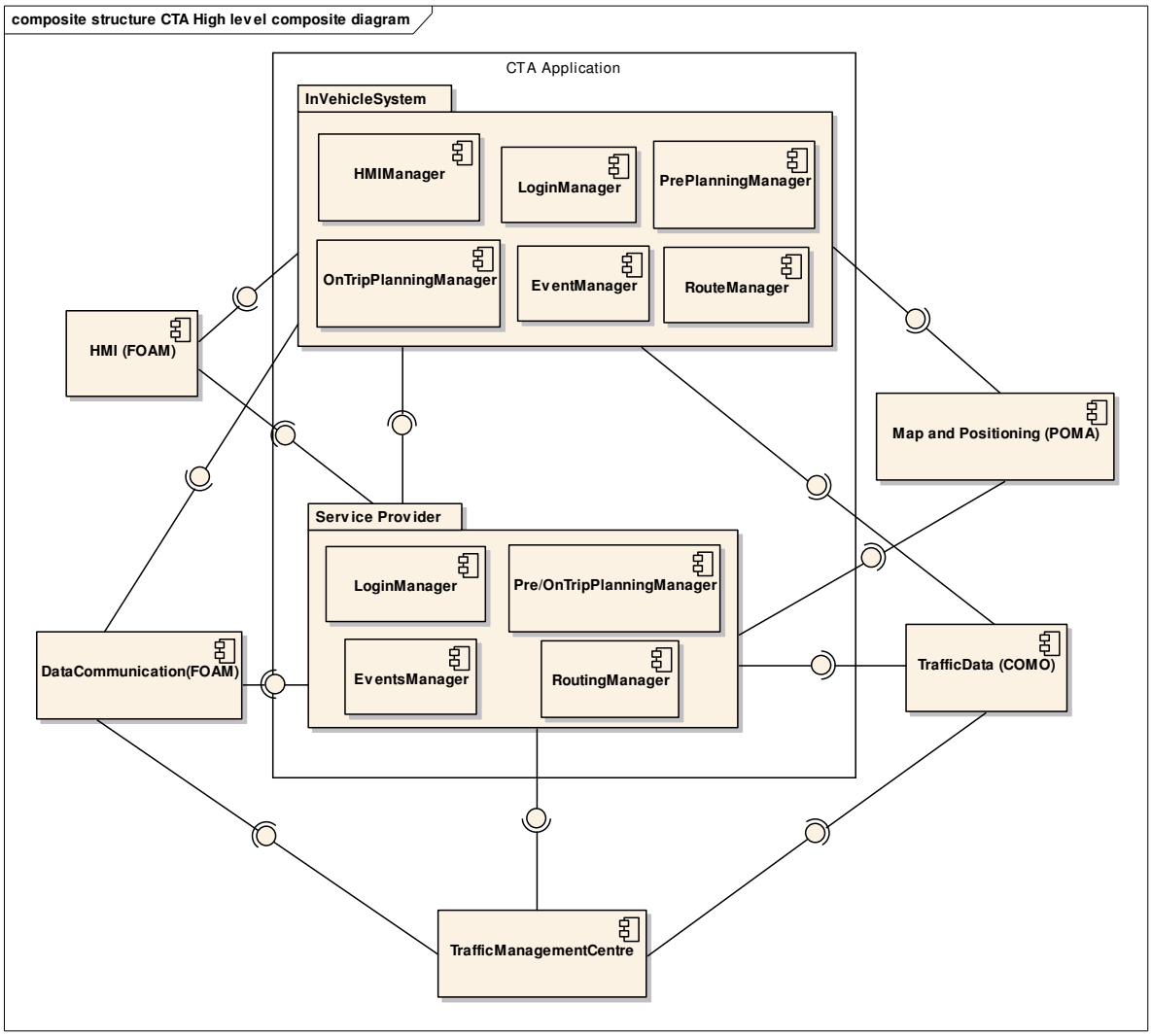

Figure 147: High level composite diagram for CTA

The CTA application is composed of two parts. One part runs on the vehicles and the other part runs on one or more service providers.

The CTA application on the vehicles is composed of the following six components:

1. **HMI manager**: this component collects and synchronizes all signals coming from other components in order to use the single canvas provided by FOAM technology sub-project;
2. **PrePlanning manager**: this component deals with the pre-planning service allowing the driver to configure the travel parameters, e.g. origin, destination, departure time, etc.;

3. **OnTripPlanning manager**: this component deals with the onTrip-planning service allowing the driver to modify one or more parameters of a planned travel or to plan a travel just before starting to drive;
4. **Event manager**: this component receives the new traffic event from a connected service provider and shows them to the driver (eg. updated map, text to speech, etc.);
5. **Route manager**: this component checks periodically the position of the vehicle, using the features provided by the POMA technology sub-project, and warns the driver if they are out of the planned route.

The CTA application on each service provider involved in the CTA environment is composed of the following four components:

1. **Pre/OnTripPlannig manager**: this component manages the PrePlanning and OnTrip Planning configuration phases providing the drivers with routes and ranked trajectories;
2. **Events manager**: this component finds out and elaborates new traffic events and sends event report to all those vehicles running in the area related to the event.
3. **Routing manager**: this component provides vehicle with the routing engine.

7.4.6 Deployment model

See Figure 147.

7.5 Enhanced driver awareness

The "Enhanced Driver Awareness" (EDA) application and its main services are introduced in this sub-section. Further details of the EDA application can be found in the D.CINT.3.2 "Architecture Specification" document.

7.5.1 Overview

The EDA application consists of two main services/sub applications that will provide awareness to travellers when they are driving vehicles as part of the journeys and to drivers of other vehicles, e.g. heavy goods vehicles, but not public transport and emergency service vehicles. Information may be sent to a single driver, or a group of drivers. The grouping of drivers may be by geographic location, e.g. a particular portion of the inter-urban road network, or some other parameter, e.g. vehicle type. The services provided by each of the EDA sub applications are as follows:

Driving advice: provides the driver with information about driving conditions (speed limit, hazard information) for the part of the road network that is immediately in the vicinity of the vehicle's current, once position and trajectory have been taken into consideration. The intention is to give drivers advanced warning of any changes to the conditions under which they are currently driving, e.g. changes in weather conditions, road conditions and speed limits and headway, plus advanced notification of traffic queues, whether they are a product of the current traffic conditions, or due to an incident of some type. Speed warning is used as the example service since the delivery of other information such as hazard information is directly analogous to the delivery of speed information. Expanding the project to accommodate all information types introduces complexity for no

learning gain.

Ghost driver detection and management: Irregular event management enables the reporting of exceptional circumstances directly to the driver; this might include emergency braking of a vehicle ahead or perhaps an upcoming traffic incident. This is exemplified by ghost driver detection and management which enables ghost drivers to be detected either by road-side units, vehicles, or their drivers. Once detected traffic managers can initiate the appropriate action to warn approaching vehicles and the recovery of the ghost driving vehicle by the emergency services. Warnings of "Ghost driver ahead" can also be provided directly to approaching vehicles.

The advice is provided to drivers either directly from the service centre to units in the vehicles, or through road-side units. These can again provide the information to in-vehicle units, or through road-side displays.

Main use cases and system boundary

The UML use case model with the system boundary for EDA is shown in Figure 148.

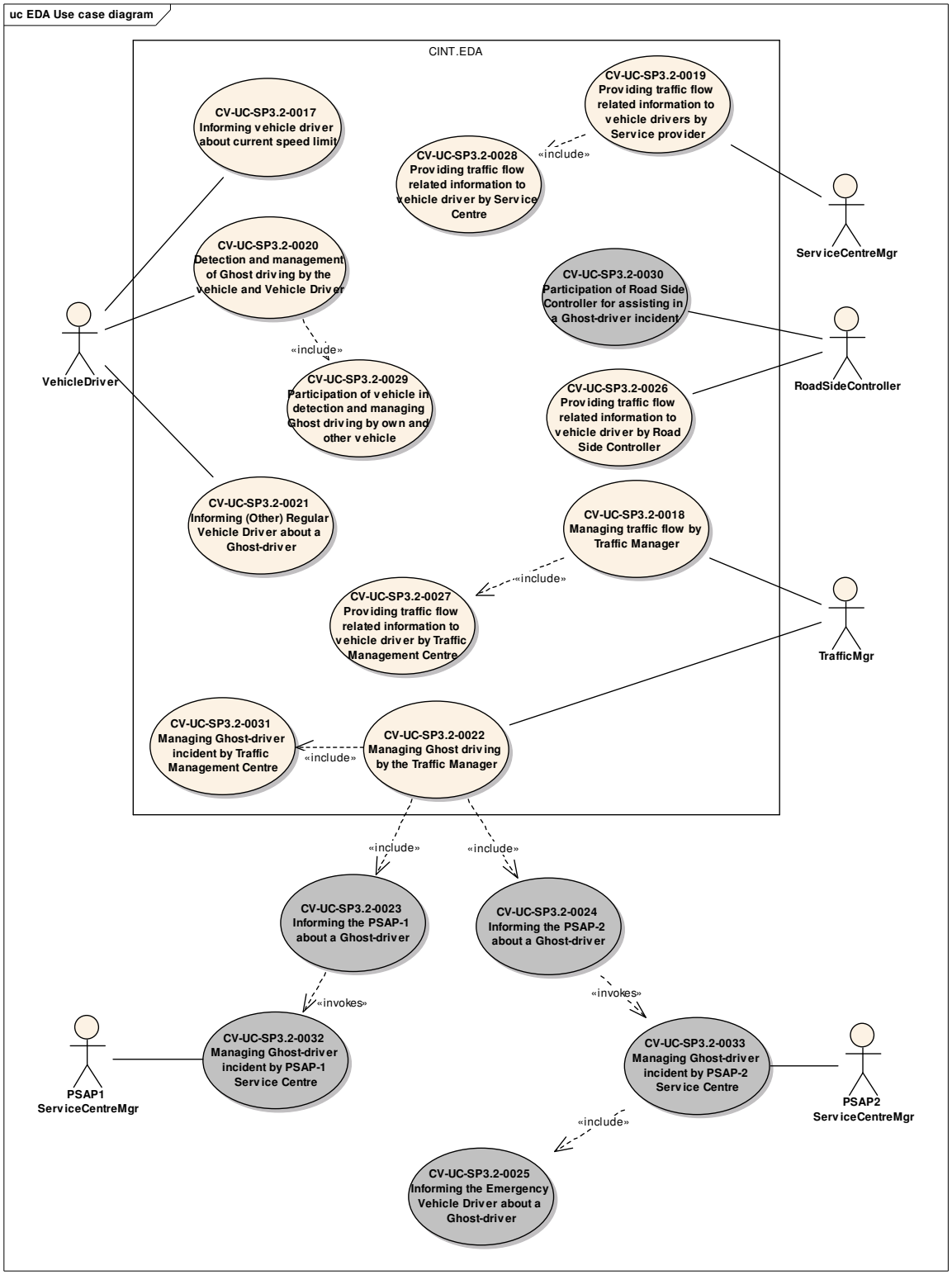

Figure 148: EDA use case model with system boundary

The use cases are as follows (for elaborated descriptions of the use cases see D. CINT D2.1):

Informing vehicle driver about current speed limit: The goal of this use case is to encourage the driver to behave lawful and adaptable to the current speed limit rule, plus the minimum headway coming with this speed limit. It is intended to tune the driving to achieve optimal, i.e. harmonized and fluid, traffic flow.

Managing traffic flow by traffic manager: The goal of this use case is to provide required information and tools for a traffic manager to optimize, i.e. harmonize and keep fluid, the traffic flow in a specific area

Providing traffic flow related information to vehicle drivers by service provider: The goal with this use case is to open an information channel from one or several service providers to the vehicle driver for specific, high value services and information which affect the driving time and the traffic flow.

Detection and management of ghost driving by the vehicle and vehicle driver: The goal of this use case is to prevent, detect and manage ghost driving by informing the vehicle driver and other actors with relevant information

Informing regular vehicle driver about a ghost driver: The goal of this use case is to inform other vehicle drivers approaching a ghost driver and using the vehicle as a dispatcher of the incident to relevant actors.

Managing ghost driving by the traffic manager: The goal of this use case is to provide relevant information and tools for a traffic manager to solve the problem situation which is associated with ghost driving.

Informing the PSAP-1 about a ghost driver: The goal of this use case is to alarm PSAP-1 for ghost driver.

Informing the PSAP-2 about a ghost driver: The goal of this use case is to react in time for saving lives due to high risks for accident assigned to ghost driving.

Informing the emergency vehicle driver about a ghost driver: The goal of this use case is to make it easier for PSAP-2 to reach the risk zone and perform necessary actions.

Providing traffic flow related information to vehicle driver by road-side controller: The goal of this use case is provide required information and tools for a road-side controller to broadcast dynamic or static traffic control information to the approaching vehicles.

Providing traffic flow related information to vehicle driver by traffic management centre: The goal of this use case is to create a direct link between traffic supervisors and the involved vehicle drivers in the daily traffic to maximize recourse, e.g. roads, utilization for efficient flow in the traffic.

Providing traffic flow related information to vehicle driver by service centre: The goal with this use case is to provide access to specific information by the vehicle driver or other actors produced by a specialized service centre.

Participation of vehicle in detection and managing ghost driving by own and other vehicle: The goal of this use case is to prevent, detect and manage ghost driving by informing the vehicle driver and other actors with relevant information.

Participation of road-side controller for assisting in a ghost -driver incident: The goal of this use case is to specify the role and participation of a road-side controller in a ghost -driver situation.

Managing ghost -driver incident by traffic management centre: The goal of this use case is to specify the information flow to and from traffic management centre as the reaction to ghost -driver incident.

Managing ghost -driver incident by PSAP-1 service centre: The goal of this use case is to specify the information flow to and from a PSAP-1 service centre in the case of ghost driving.

Managing ghost -driver incident by PSAP-2 service centre: The goal of this use case is to manage and guide the emergency vehicles needed to abate the event.

The actors and their needs and responsibilities are described in the following:

"TrafficMgr: This is a human entity that manages the operation of the CINT (and other) applications forming the traffic management system located in the traffic management centre (TMC) that is responsible for the inter-urban road network. The traffic manager is able to manage how the applications in the TMC operate and the information that is made available to vehicle drivers and travellers. Additionally the traffic manager can decide on the way that vehicles are able to use the network, e.g. OPEN/CLOSE lanes, and set speed limits, etc.

ServiceCentreMgr: This is a human entity that manages the operation of a service centre system. The service centre manager can regulate how the responses to requests from the vehicle drivers (or travellers) are provided by the CINT applications running in the service centre. The manager can also provide data for use by these applications.

PSAP1ServiceCentreMgr: A service centre manager responsible for access point 1. This access point can be used by travellers and other people to report incidents and other forms of emergency. It can also provide information to travellers and others about incidents that have been detected by other means.

PSAP2ServiceCentreMgr: The same as the PSAP1 service centre manager. The second access point is included to indicate that all these services do not have to be provided by the same service provider and to enable the exchange of information and data between different instances to be shown.

RoadSideController: This represents one or more physical entities that are situated at or close to one or more points at the side of the inter-urban road network. These entities are capable of displaying information when requested to do so by applications within the system. The displayed information may comprise either an indication that a lane is OPEN or CLOSED (international symbol for CLOSED is a red "X"), a speed restriction, or a message. With the exception of the CLOSED indication, all other outputs may be either compulsory (internationally indicated by a red ring around the display and/or flashing red lights) or advisory (internationally indicated by flashing yellow lights). A message will be a text string advising vehicle drivers of a particular situation that exists ahead of them in the inter-urban road network.

VehicleDriver: This is the human entity that controls a licensed vehicle operating on the inter-urban road network and is able to interact with CVIS applications in the vehicle. Operators of private, freight, public transport and emergency services vehicles shall be

included. The entity shall originate driver requests to, and receives driver information from the system. It shall be possible to output different information to each type of driver without the other types of driver seeing the outputs.

7.5.2 Application programming interface

This section describes that interfaces that the EDA applications will have with entities in the outside world.

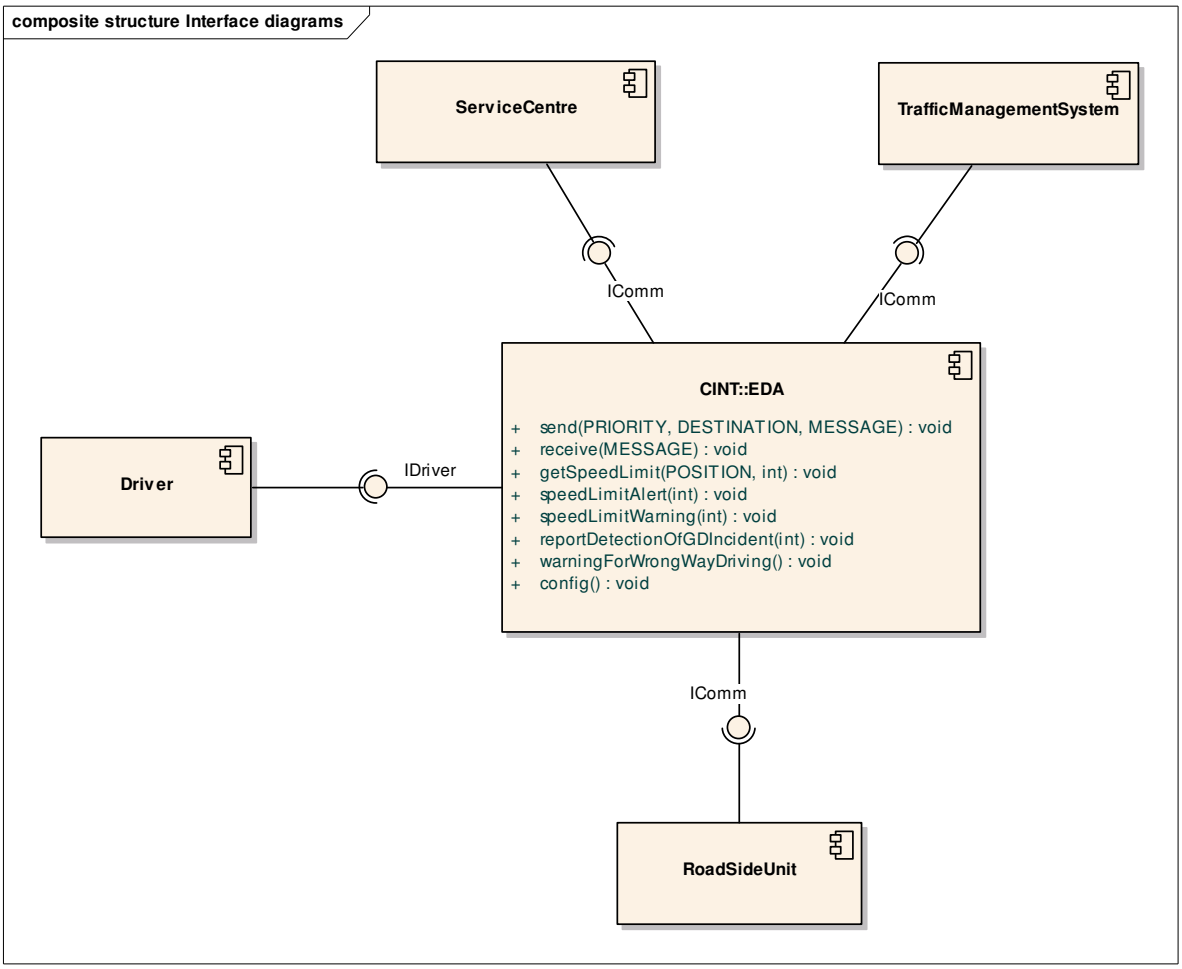

Figure 149: EDA external interface

The top level part of the EDA application interacts with the driver and legacy systems like service centre, traffic management system and road-side unit. It is assumed that they will already provide a channel for communication in the CVIS network. As it is common to define a communication channel into a communication and application protocol, the specified interface in EDA is mainly referring to the application protocol. This means that the encoding and decoding of messages aimed for EDA will be handled by the provided feature.

The interface with the driver consists of a number of features which are mainly related to speed limit awareness and detection of ghost driving.

7.5.3 Information model

Some main information objects of the EDA application is depicted in Figure 150.

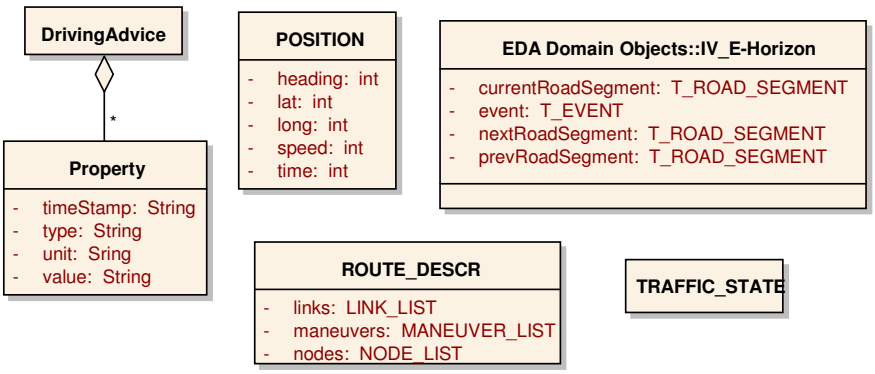

Figure 150: EDA information model

7.5.4 Interaction model

The following sequence diagrams show example interactions of the two sub applications of the EDA (driving advice sub application and ghost driver detection).

The following sequence diagram shows how current speed limit information will be brought to the vehicle so that it can be displayed.

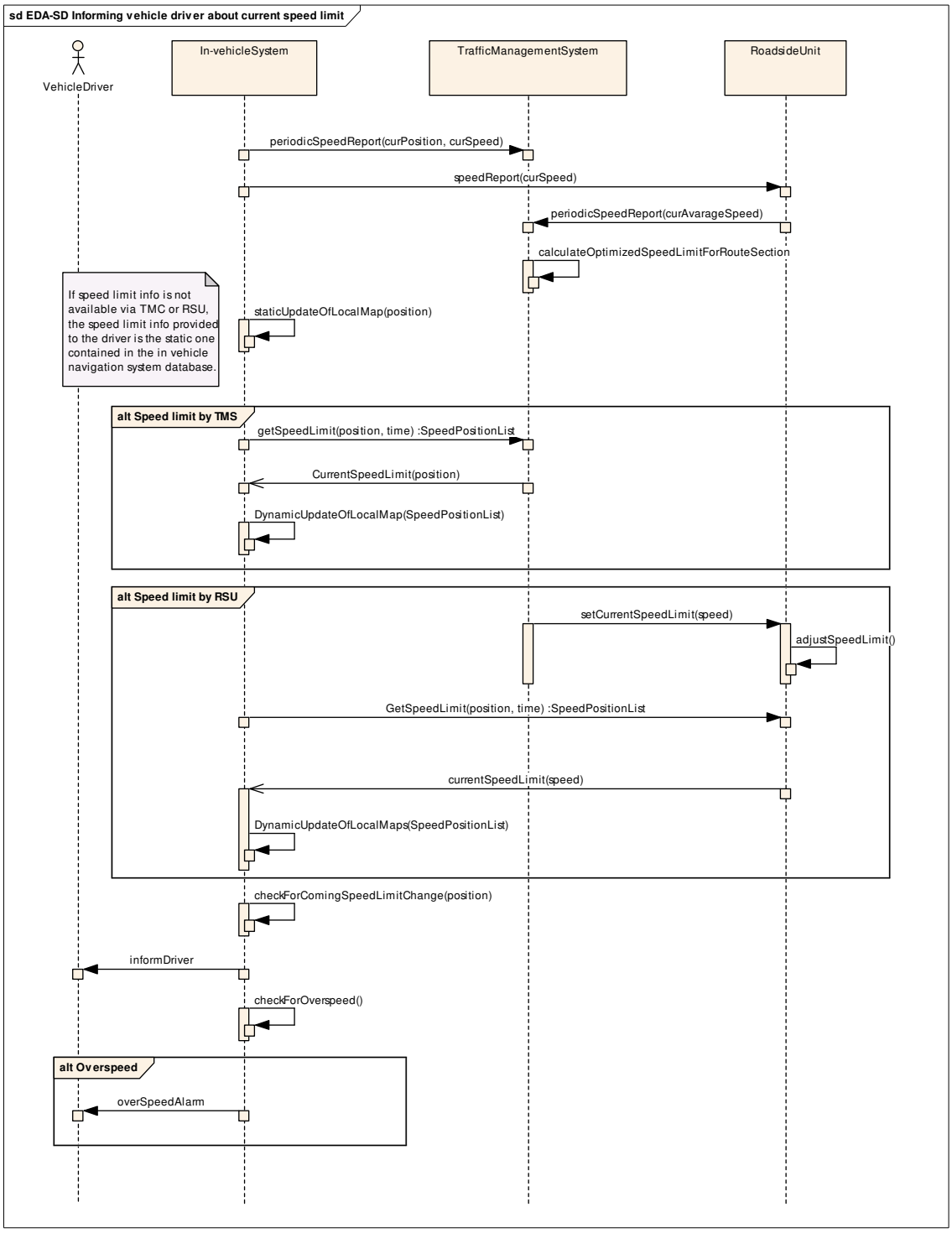

Figure 151: Sequence diagram for informing the driver about the current speed limit

The next sequence diagram shows detection of a ghost driver by the vehicle and the vehicle driver.

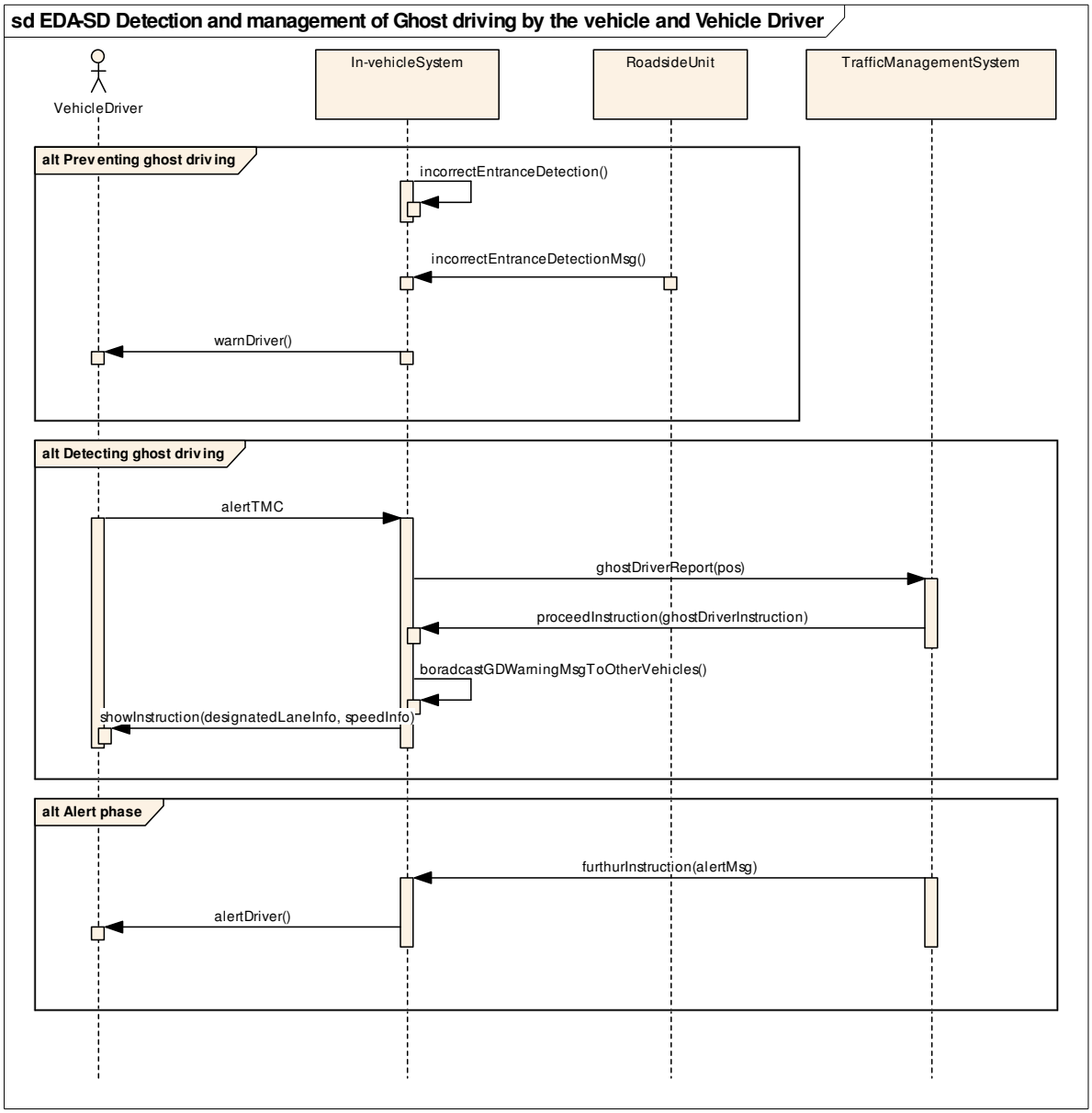

Figure 152: Sequence diagram for ghost driver detection by vehicle

The EDA service for detection of a ghost driver by the vehicle and vehicle driver involves the in-vehicle system, road-side unit, traffic management system and vehicle driver entities. The three stages which occur chronologically can be identified as follows:

Preventing stage,

Detecting stage,

Alerting stage.

Further descriptions of interactions with respect to the ghost driver sub application are provided in D.CINT.3.2.

7.5.5 High level composite architecture

The high-level composite architecture diagram for the EDA part of CINT is shown in Figure 153. It includes the interfaces of the EDA application with other CVIS services.

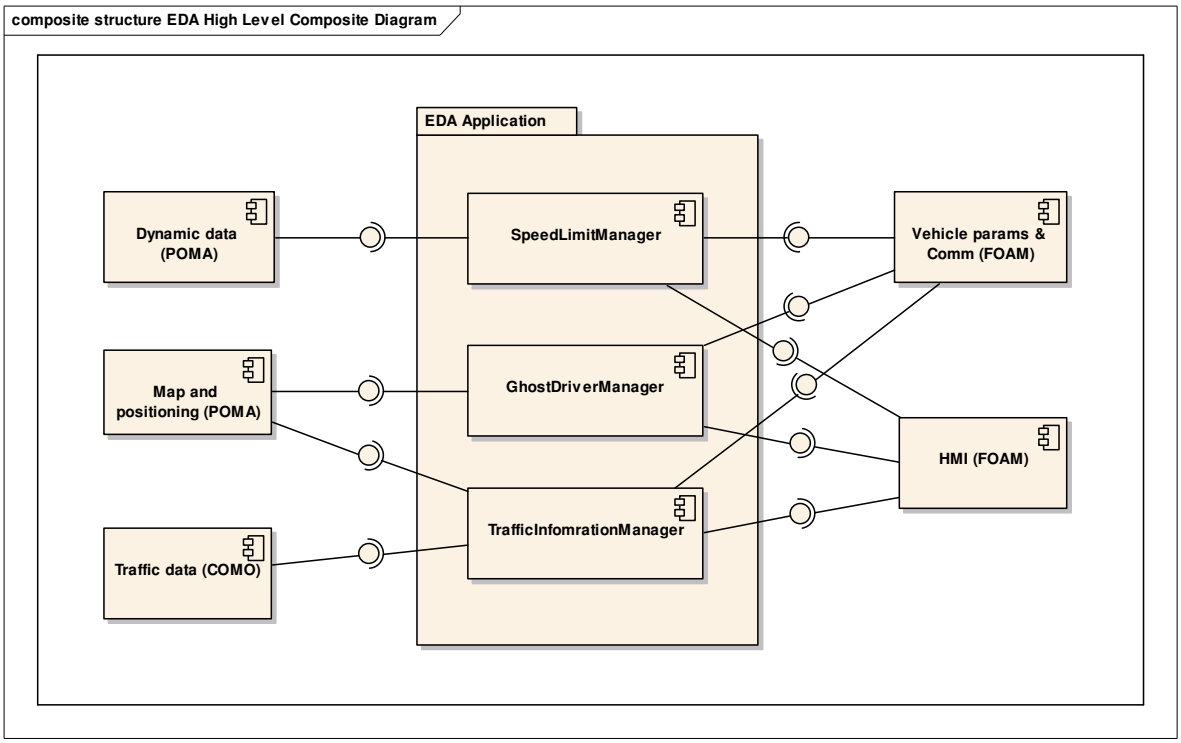

Figure 153: High level composite diagram for the EDA application

7.5.6 Deployment model

Figure 154 shows how the EDA application with its two sub applications will be deployed in a CVIS physical environment.

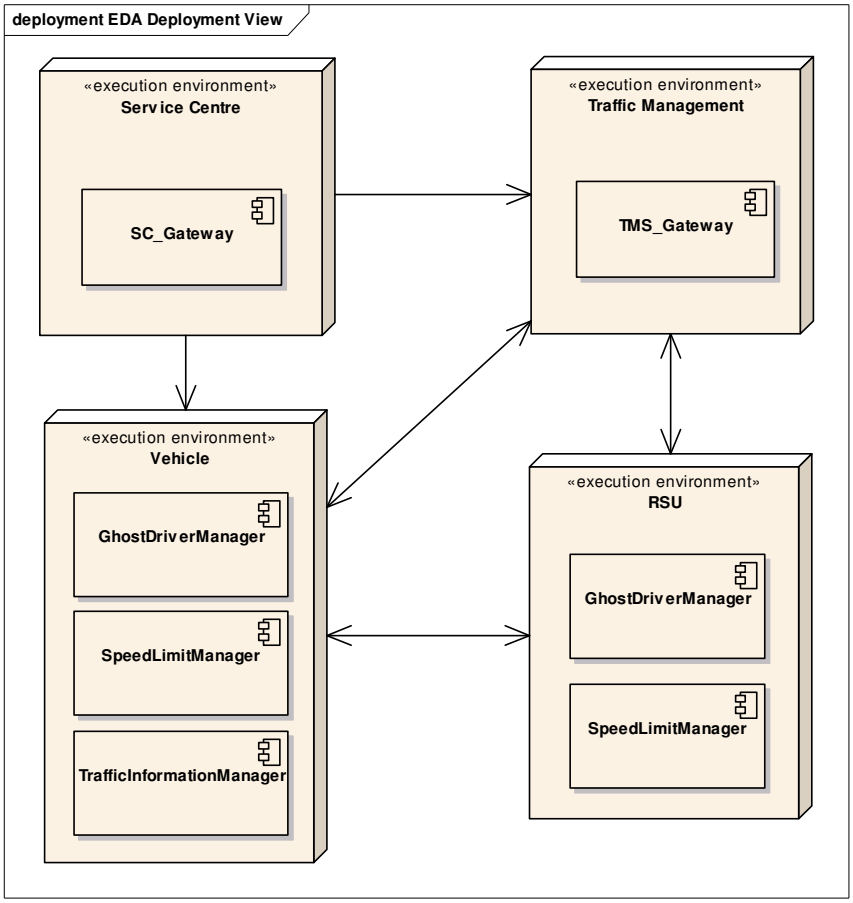

Figure 154: Deployment diagram for the EDA application

7.6 Information application

The information application and its main services are introduced in this sub-section. Further details of the information application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.6.1 Overview

A driver can subscribe a service to receive information about traffic states, incidents information and alternative route options.

A driver who subscribed is informed about the traffic and incident situation of a certain urban network and receive individual information pre-trip an on-trip. He has the possibility to react on congestions and incidents by choosing different routes, according their knowledge of the urban network, or change his intended starting time.

By continuously monitoring, analyzing and predicting the road network state, high quality traffic information is collected. Based on the content of the information, the driver's request,

location and direction of his vehicle, information is selected on relevance, broadcasted to the on-board computer and presented to the driver via a HMI.

The activity flow is as follows. A driver selects its destination in the HMI. A route to the destination is calculated. The driver can select to receive on-trip information about traffic and incidents situations. When the destination is selected in advance, the driver can also select to receive pre-trip information. The information applications running on the vehicle CVIS host system will send information requests to the urban centre and road-side units along the trip. The received information from the urban centre and road-side units is filtered on relevance and presented on the HMI.

Main use cases and system boundary

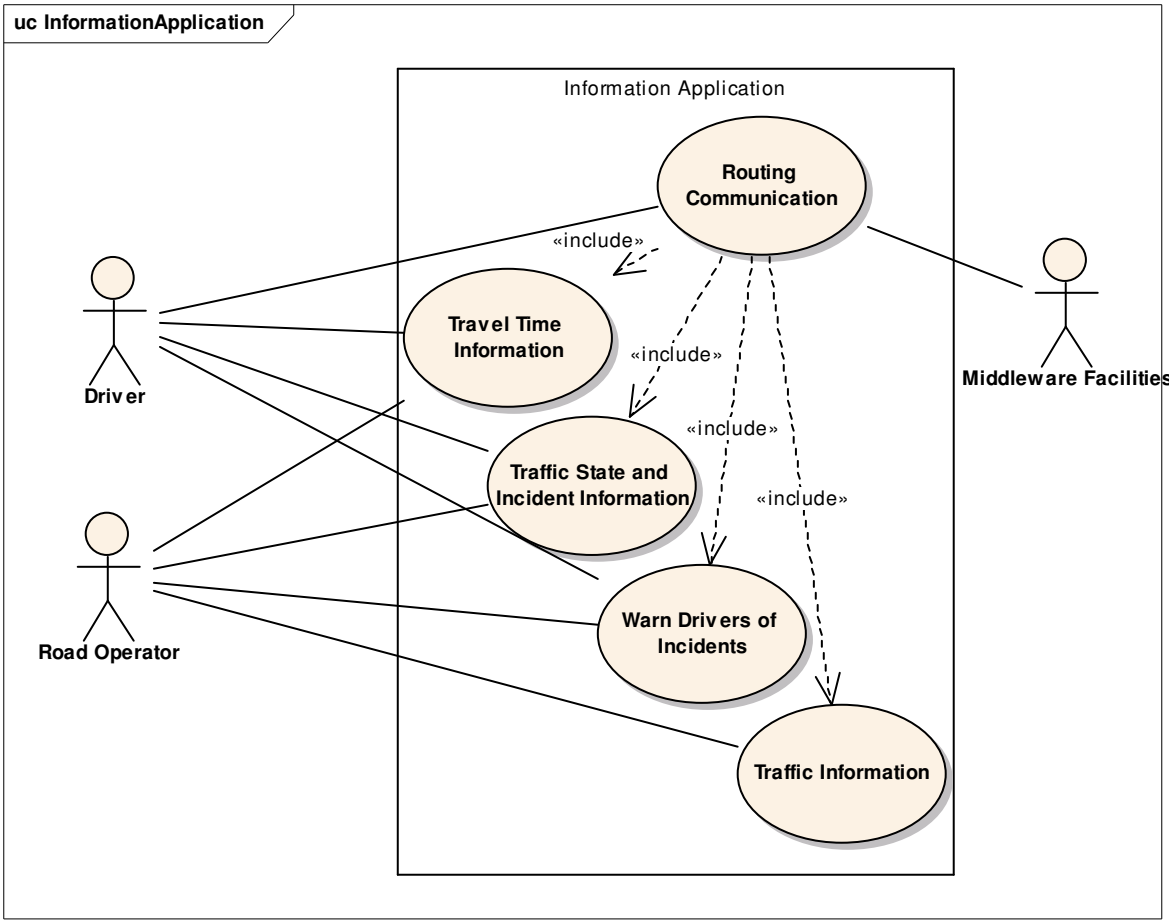

Figure 155: Use case model with system boundary

The use cases are as follows:

Area routing communication: The goal of this use case is to provide: i) the communication of the route options to the driver and ii) the computation of route options based at infrastructure side based on the current locally available traffic information. When a congestion or a potential congestion is identified, the information is (back/fore) propagated to other local units, e.g. on the upstream/downstream, or at central level. The information can be communicated via local unit or by central system. The vehicle informs the local unit of its position, its destination, its vehicle type (it is possible to generate routes option specific for private, public transport, commercial and heavy vehicles) and its

possible paths. The route options or related information are communicated to the vehicles either by broadcasting, via local communication or any/unicast. By processing traffic data, identifying congestion events and compute route options at local level the reaction and processing time of the CVIS routing service are optimized. Furthermore local strategies can be considered in the computation of local routing advices.

Local and area traffic information distribution and enrichment: The goal of this use case is the distribution of network and local traffic situation and short and medium time prediction at network, area and local level. The information is supposed to come from higher level and to be eventually enriched with more detailed information, both from vehicle (private/commercial/public) and from conventional infrastructure based sensors. The information could be locally distributed without elaboration or enriched with local status and prediction. A major task is to covey and resolve conflicting information that may come from higher level systems (as traffic management and route guidance systems).

Generate and provide traffic state and incident information to individual vehicles: The goal of this use case is to provide consistent high quality traffic information as on onboard service to the road user. The road user can request information about incidents and traffic state in certain urban networks or parts of urban networks. The information service is not connected with a routing service is therefore addressing certain user groups. Only informing the drivers on traffic conditions and incidents is a service which is especially useful to road users who are well known with the network (commuters). By warning the drivers of congestions and incidents shortly after the detection the road safety can be improved. In addition the drivers have the possibility to use alternative routes according to their knowledge of the network.

Warn drivers of incidents in the urban network: The goal of this use case is to inform drivers about incidents in the urban network. The drivers are aware of the incident and are able to adjust their driving behaviour. Safety is increased.

Travel time per destination information to driver: Actors involved are the driver of the CVIS equipped vehicle, traffic management centres that provide the traffic information and the road-side units in the CVIS network that distribute local traffic information.

Driver: The driver wants to travel through the urban network in a comfortable, safe and efficient way. He is interested in being supported by dynamic information and navigation systems.

Road Operator: The road operator is the organization responsible for maintaining the roads and managing the traffic. The road operator wants to improve traffic control and management by informing the drivers about the traffic and incident situations.

System Operator: The system operator is the organization responsible for maintaining the road-side equipment. The system operator wants to improve their equipment by implementing CVIS support.

7.6.2 Application programming interface

For an application the main external interface is the interface provided to the end user. The end user API for the priority application is shown in Figure 156.

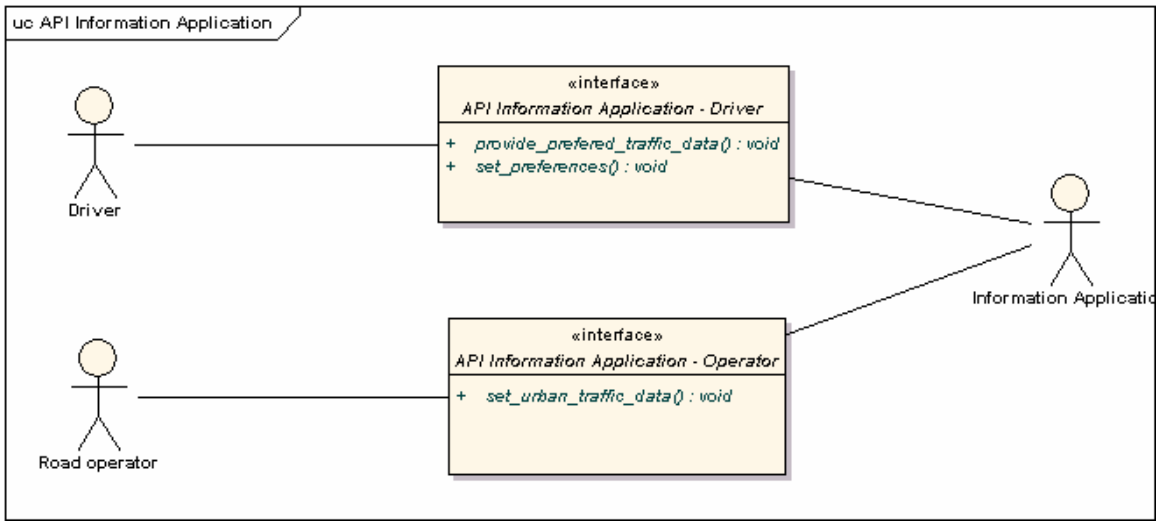

Figure 156: API information application

The behavioural aspect of the interface is shown in Figure 157.

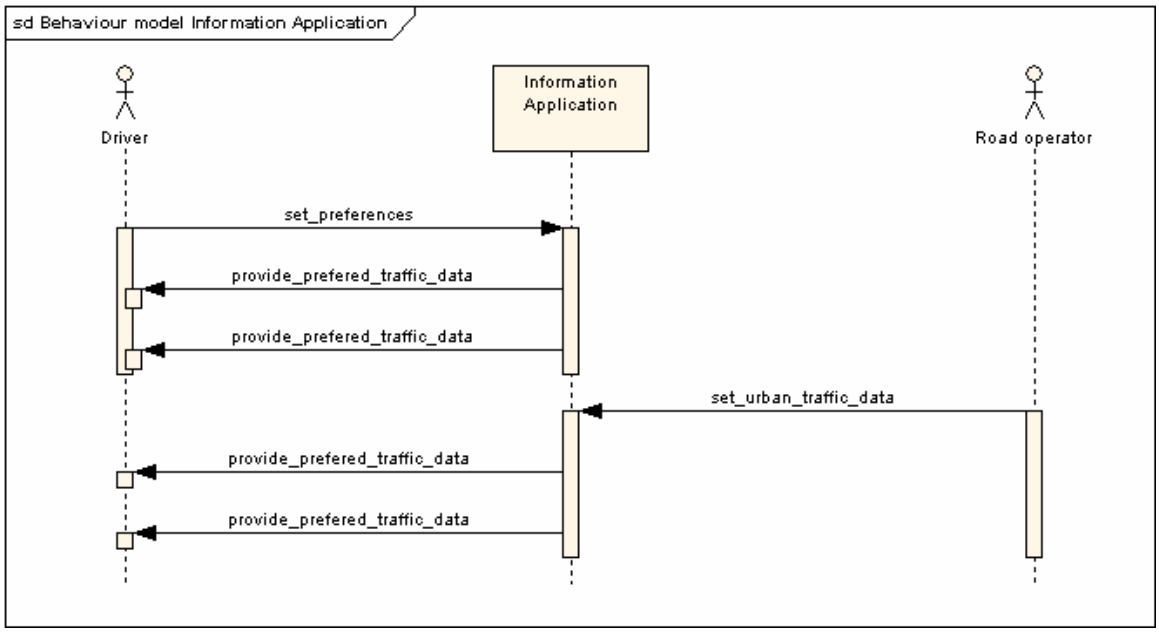

Figure 157: Behaviour model information application

7.6.3 Information model

This section provides the specification of the information model. The information model identifies and defines the main concepts of the information application domain. The concepts are specified in terms of their types using UML class diagram as shown in Figure 158.

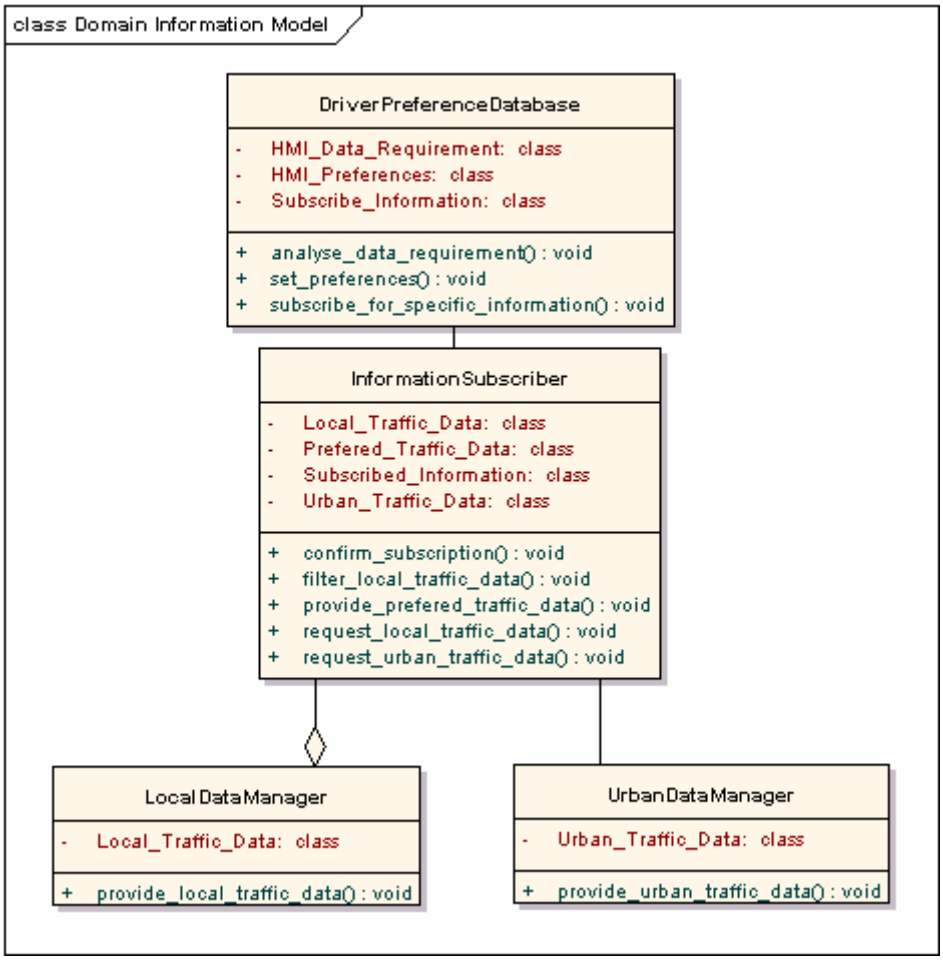

Figure 158: Domain information model information application

7.6.4 Interaction model

In detail the sequence of events of the information application is as follows:

The driver starts its CVIS equipped vehicle.

The driver selects his destination and preferences on the HMI.

The HMI stores the selected preferences into a "Driver Preference" database.

The "Driver Preference" database will make a request to the information application for the specific information.

The information application periodicals send requests for information to the urban centre and the available road-side units along the trip based on the specific request. The received information is filtered according the specific request.

Only the preferred information is shown on the HMI.

This main interaction is depicted in the process model of Figure 159.

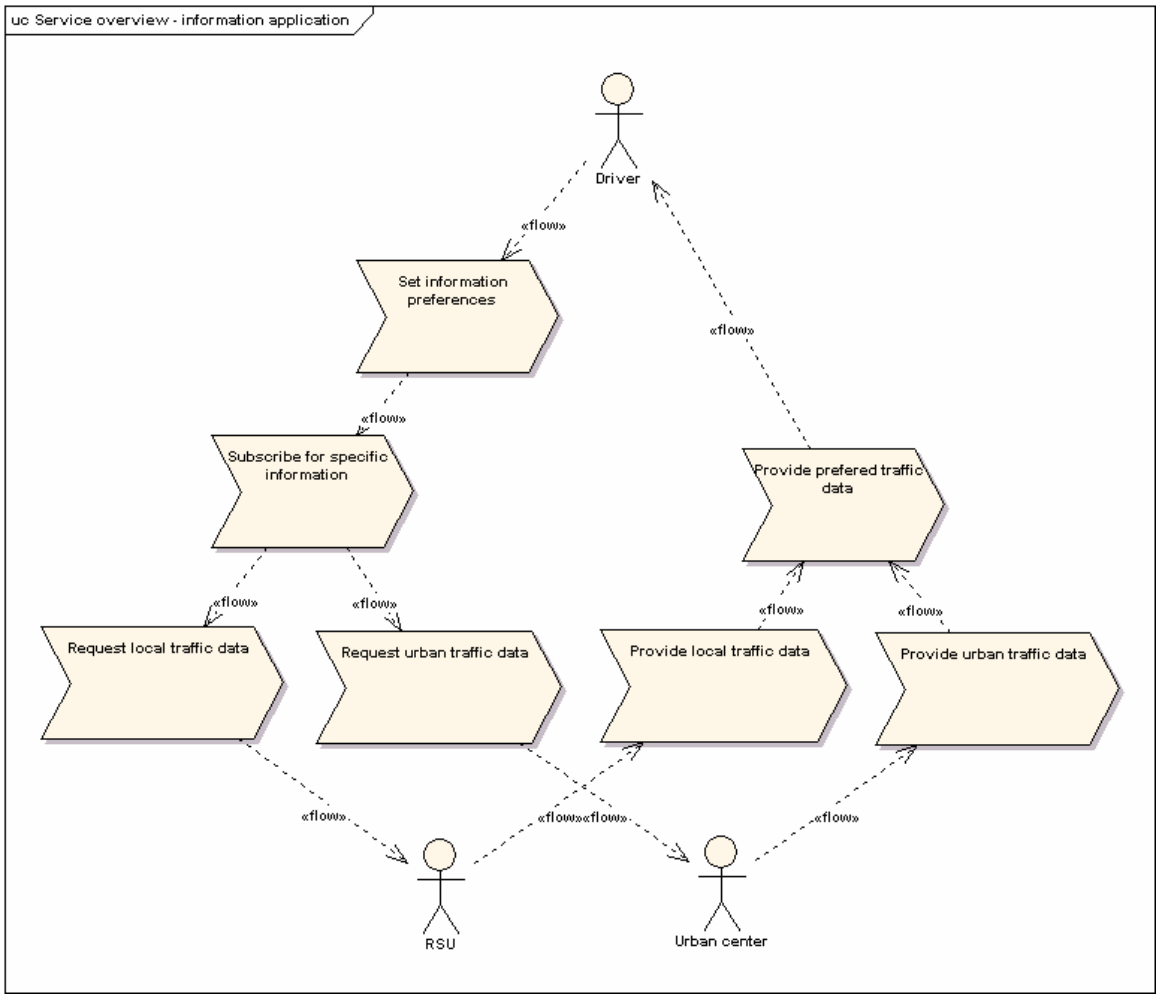

Figure 159: Reference service process information application

A detailed scenario description is provided below:

A CVIS enabled urban centre registers its services, e.g. information application, and properties, e.g. identifier and communication handle, to the "Distributed Directory Service" (DDS). It can as well receive subscriptions from the vehicle applications.

A CVIS enabled road-side unit registers its services, e.g. information application, and properties, e.g. identifier, location, reception range and communication handle, to the DDS. It can as well receive subscriptions from the vehicle applications.

Road-side host activation: configuration files of the road-side services are sent to the DDS. From there they are globally available within CVIS.

When a driver starts his CVIS enabled vehicle he has to authenticate himself to the CVIS system.

Vehicle host start-up: the CVIS equipment is started.

"Authentication": The driver has to authenticate himself to the CVIS system by providing some credentials, e.g. name, password, smartcard or biometric data.

Vehicle host activation: [I am alive message] is send to the "Host Management Centre"

(HMC).

Configuration of CVIS vehicle host:

User profile activation: after authentication, the HMC will look up the stored end user profile (if it exists). Such a user profile, e.g. myCVIS, consists of all mandatory public services and optional free or paid services previously selected by the user.

Change user profile: after user profile activation, the user will be given the opportunity to change his profile by adding or deleting (available) optional services from a list provided form the DDS.

Note that it only involves the services which are available within a specified range of the vehicle, most likely the fuel range.

Synchronization: the HMC subscribes the user to the selected services and verifies, through the remote management protocol, whether the services belonging to the user profile (or the right version of them) are actually present on the CVIS vehicle host. If not, they will be provisioned by accessing the DDS including the properties (see 1) of the partnering road-side units.

Note due to the so called "Selection Criteria" only the services which are available along the route (in case the route is known) or within a range of X kilometres.

On the HMI the driver can set the preferences for receiving traffic information on-trip and pre-trip. These preferences are stored in the "Driver Preference Database".

When the driver selects his destination on the HMI a route is calculated. Depending on the preferences a subscription is made for specific information.

While driving, the CVIS vehicle host needs to check regularly if the present services are still sufficient and/or valid. This can either be done:

- Time wise

- Location wise

The vehicle continuously monitors its location and the need to start communicating to a CVIS road-side host, e.g. the information application.

The information application will periodical sends information requests to the urban centre and the road-side units on the trip.

The provided data from the road-side units is filtered and only the preferred traffic data is shown to the driver on the HMI.

7.6.5 High level composite architecture

The high level composite architecture of the information application is shown in Figure 160.

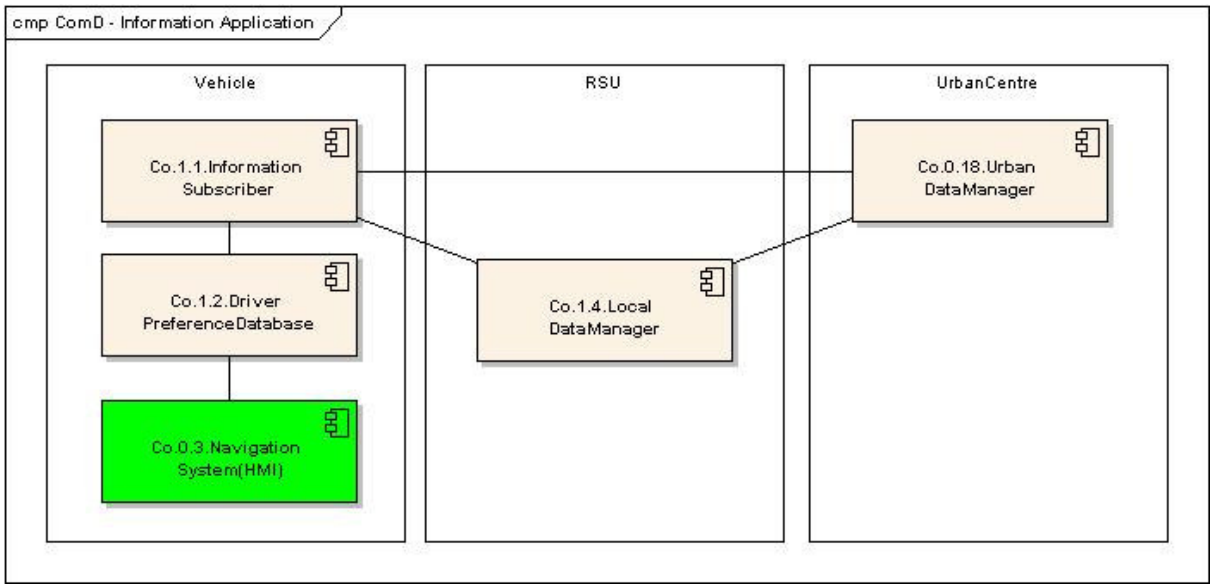

Figure 160: Component diagram priority application

Co.1.1.Information subscriber*:* subscribes for specific traffic information based on the preferences of the driver. The information comes from an urban or local level and will be pushed to the in-car CVIS system based on the subscription.

Co.1.2.Driver preference database: contains algorithms to determine which information is relevant for the driver in the current situation. This component informs the information subscriber which types of information are useful and on the other hand presents the received data in accordance with the drivers' preferences.

Co.0.3.Navigation system (HMI): The navigation system is the HMI to the driver. Based on the input of the driver, the preferences of the driver are determined. Furthermore, the HMI presents the received data.

Co.1.4.Local data manager: The local data manager gathers, fuses, stores and distributes local traffic information.

Co.0.18.Urban data manager: The urban data manager gathers, fuses, stores and distributes urban traffic information.

7.6.6 Deployment model

This section describes the logical deployment of the priority application onto the CVIS infrastructure using UML deployment diagram as shown in Figure 161.

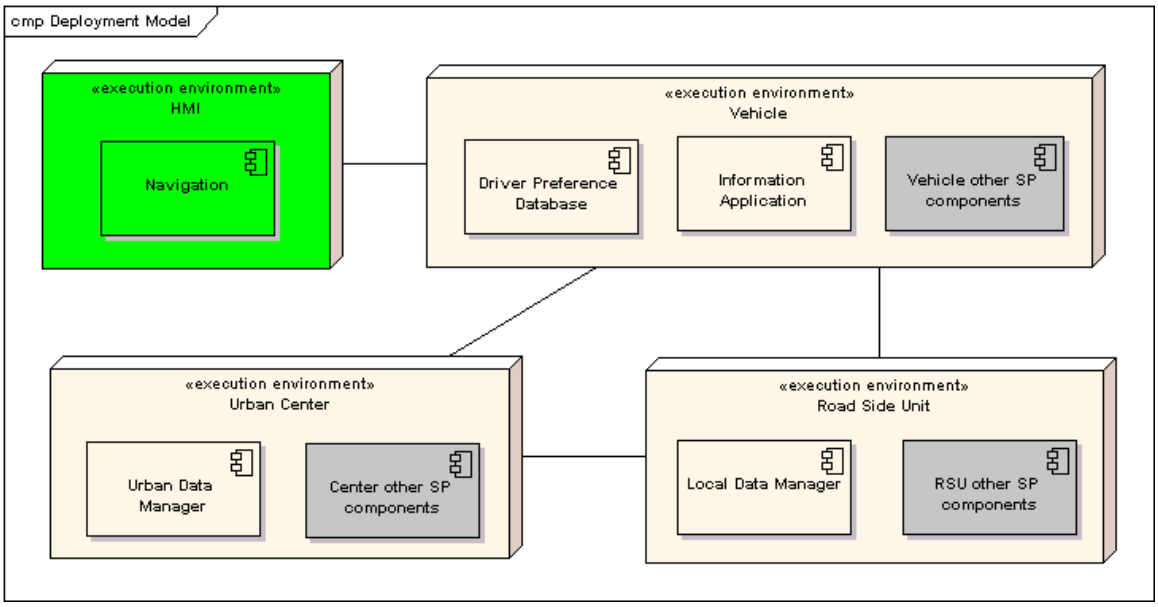

Figure 161: UML deployment diagram

7.7 Priority application

The priority application and its main services are introduced in this sub-section. Further details of the priority application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.7.1 Overview

Some vehicles deserve higher attention than others, for instance emergency vehicles, public transport vehicles, heavy trucks or trucks for DG. Public authorities and road operators should be able to configure the local traffic management setting according to their policies and rules with regard to these vehicles. In correspondence with the use case 'Request for green, the goal of the priority application is to create a vehicle requested 'green window' in a cooperative way. It aims at a more fluid (and safe) intersection crossing for the vehicle categories set by the authorities. Feedback to the driver is optional. Possible, the priority application could be extended by the speed profile application.

The activity flow is as follows. When a special category vehicle approaches a signalled intersection, the vehicle identifies itself and informs a road-side unit of its expected time of arrival at the stop line based on free flow conditions. The road-side unit gives the vehicle a priority level and integrates all green requests into the signal program. Then, the road-side unit implements the updated signal program into the traffic light controller and simultaneously sends the green planning to the special category vehicle. Optionally, the vehicle is able to calculate the desired speed to maximize efficiency.

Actors involved are the driver of the CVIS equipped vehicle in the predefined vehicle categories, fleet owners of the equipped vehicles, road operators responsible for local traffic

management and system operators owning the CVIS equipped traffic light controller and/or road-side unit.

Figure 162: Non-formal representation of the *priority application*

Main use cases and system boundary

The main use case of the priority application is 'Request for green'. In general an acknowledgement message can be returned to the requesting vehicle or a number of vehicles in a specific area However, within the field of cooperative urban applications it makes sense to integrate the priority application and its use case with services provided by other CURB applications. For instance a response to request for green may be to send a return message containing information on the planned green times and the recommended vehicle speed in order to meet the green window that is now been set up. The speed profile application provides such kind of information to vehicles in the vicinity of a controlled intersection. Therefore, integration with the speed profile application could be a logical extension of the priority application. This is indicated in the use case model shown in Figure 163.

Furthermore, one of the main aims for cooperative vehicle-infrastructure systems is to optimise traffic control systems through extensive and enriched traffic data. This aim is of particular focus in the use case 'Cooperative Traffic Control' and its corresponding application. Basically, the priority application is an example of functionality which enables cooperative traffic control. Thus it could be convenient to integrate the priority application also with the cooperative traffic control application.

The dependencies to middleware facilities are represented in the use case model using an actor.

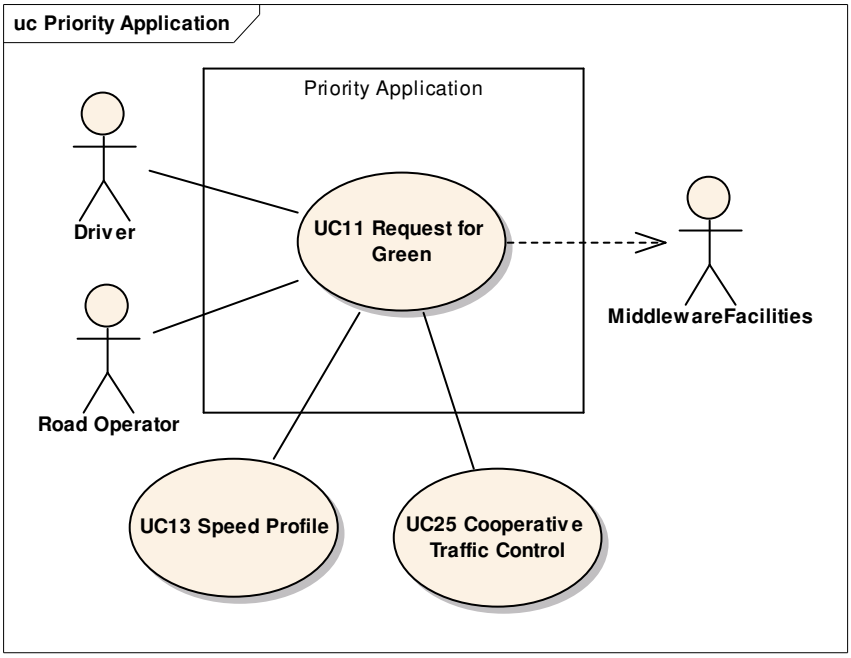

Figure 163: Use case model with system boundary

The use cases are as follows:

UC11 *Request for green*: Creation of a "requested green" in a cooperative way (targeted at special road user categories). This use case describes the creation of a, vehicle requested, green on a controlled intersection. This requested green is used to give priority to special road categories like emergency vehicles, trucks with DG, public transport etc. For those categories, it aims at a more fluid (and safe) intersection crossing. Priorities for the different categories could and should be set by authorities. A feedback loop to the drivers is optionally available.

UC13 *Speed profile*: This use-case describes an important feature to increase efficiency of an intersection and the network. For maximum effect is application should be available on all vehicles. The use-case has two main components. The first is to extend the controller algorithms to deal with more and more detailed information received from CVIS vehicles (FCD). The second main component is broadcast of speed profiles. The broadcast facility is used to broadcast speed profiles to CVIS equipped vehicles driving in a certain direction (please note that this is different from "destination"). The speed profile is a short time profile with speed advice (suggesting acceleration / deceleration or just suggesting the average approaching speed). The profile will be interpreted by the vehicles' on-board unit and if necessary presented to the driver.

UC25 *Cooperative traffic control*: The goal of this use case is to optimize traffic flows in a limited area (up to 5 intersections), based on all kind of available traffic information (esp. XFCD and road-side sensor data) and using different methods of traffic control. Within the control area, one co-operative intersection will be equipped with a Master device; the others will be treated as sub-ordinate nodes.

The actors and their needs and responsibilities are described in the following:

The private driver wants to travel through the urban network in a comfortable, safe and efficient way. He is interested in being supported by dynamic information and navigation systems.

"Road operator: Organisation responsible for maintaining the roads and managing the traffic. The road operator wants to improve traffic control and traffic management by defining and implementing cooperative control and management strategies.

Middleware facilities: Represents the CVIS basic and domain facilities (see part II of this document)

7.7.2 Application programming interface

For an application the main external interface is the interface provided to the end user. The end user API for the priority application is shown in Figure 164

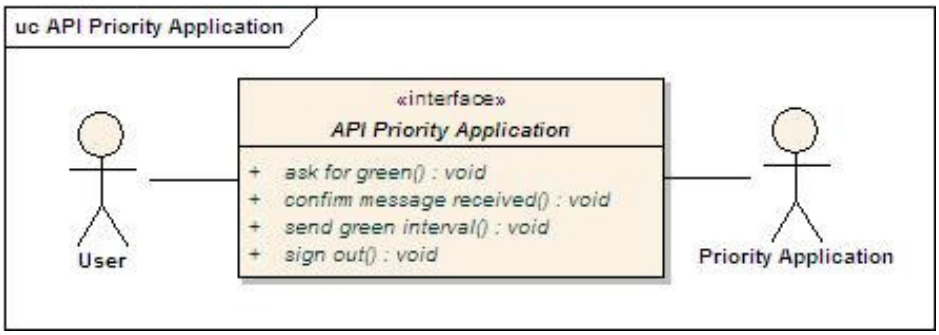

Figure 164: API priority application

The behavioural aspect of the interface is shown in Figure 165.

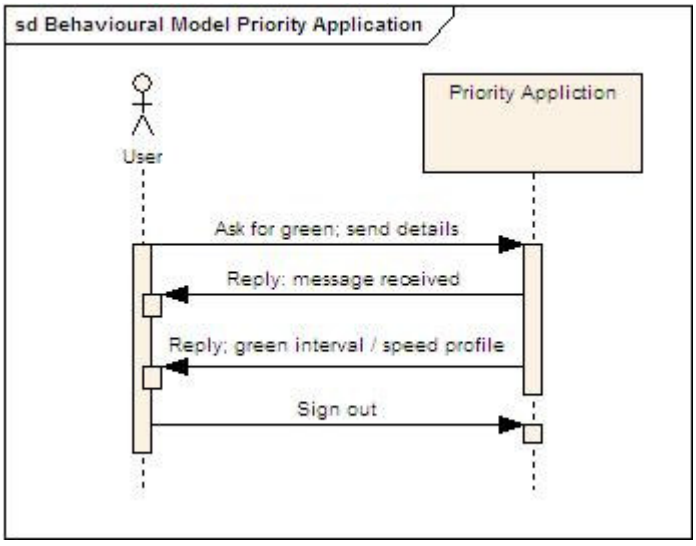

Figure 165: Behavioural model priority application

7.7.3 Information model

This section provides the specification of the information model. The information model identifies and defines the main concepts of the priority application domain. The concepts are specified in terms of their types using UML class diagram as shown in Figure 166.

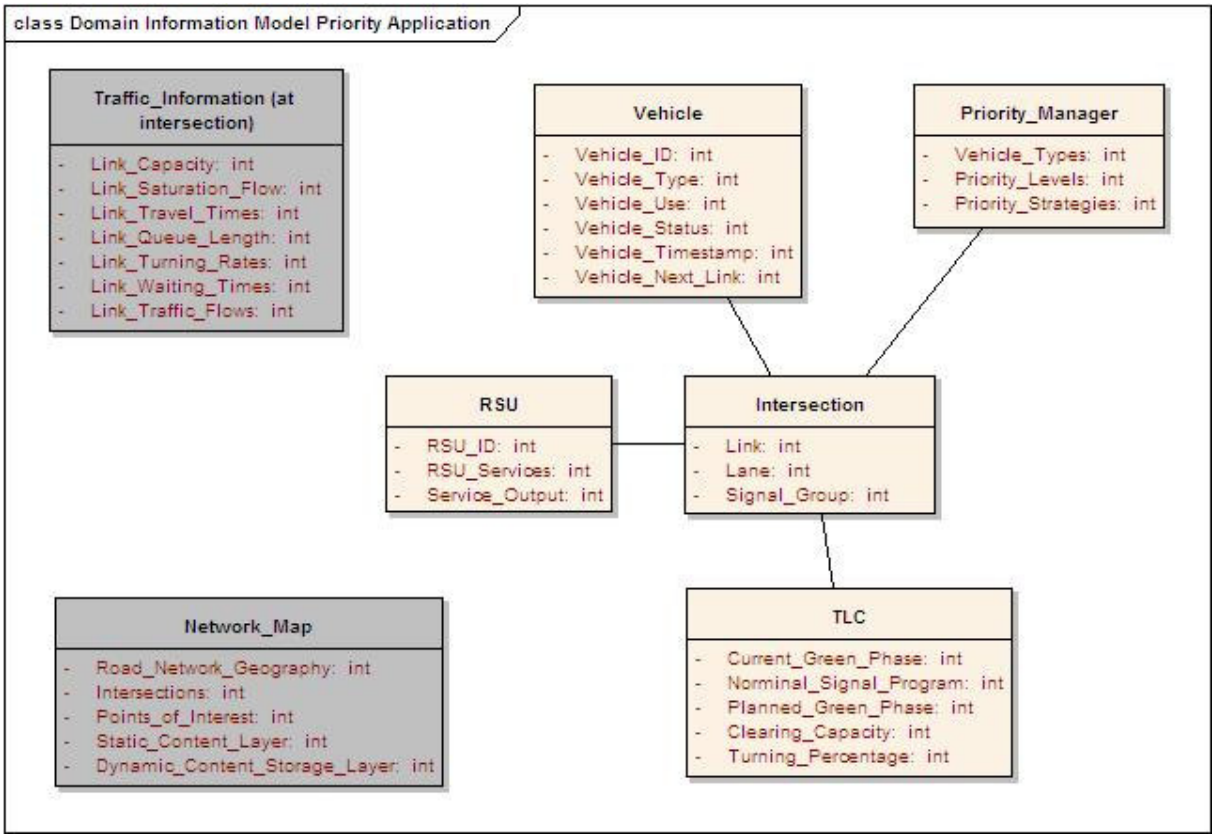

Figure 166: Domain information model priority application

7.7.4 Interaction model

This section provides specifications of main domain processes. The overall value chain of the priority application is shown in Figure 167.

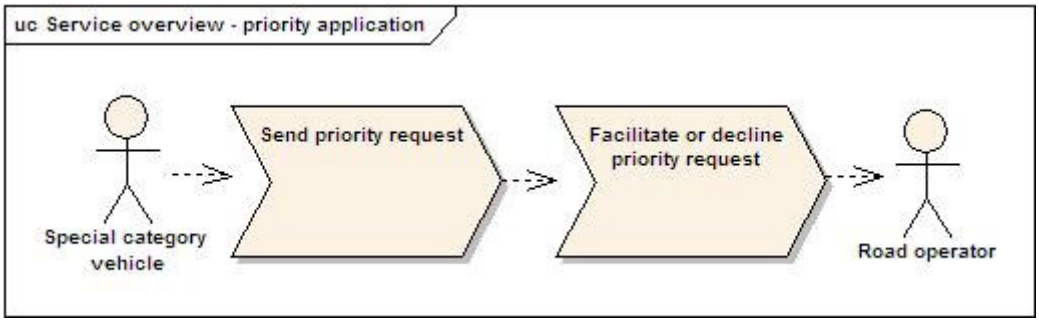

Figure 167: Reference service process priority application

The workflow of the priority application is specified using UML activity diagram as shown in Figure 168. The return message is specified as optional since a more extensive return message can be provided when integrating with for instance the speed profile application.

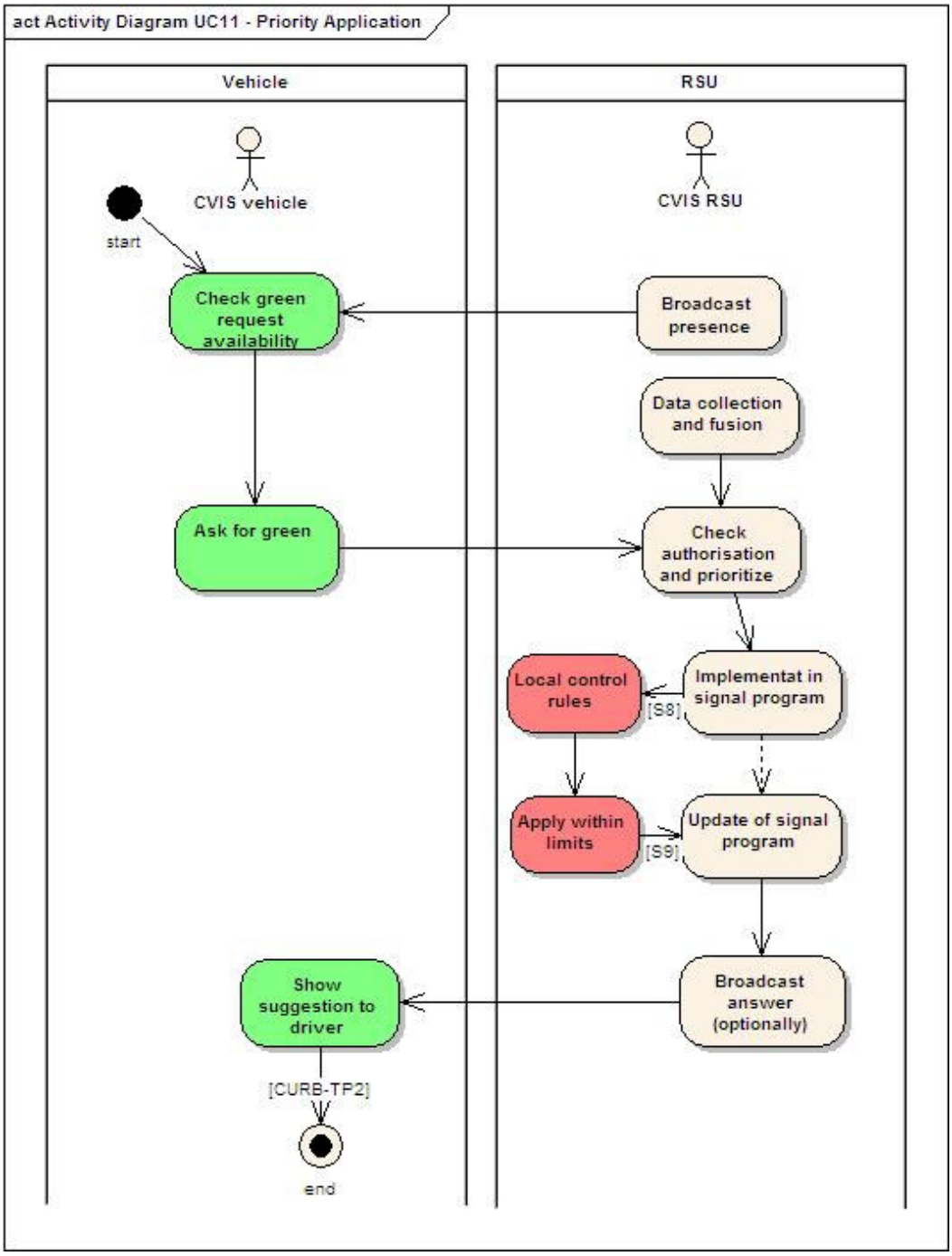

Figure 168: Activity diagram priority application

Comprehensive Scenario description

A more comprehensive scenario description including interaction with middleware facilities is specified in the following.

After successful start-up, a CVIS enabled RSU registers its services, e.g. priority application, and properties, e.g. identifier, location, reception range and communication handle, to the "Distributed Directory Service" (DDS). It can as well subscribe to messages originating from the vehicle part of the application. This allows for fast communication setup when a vehicle enters the reception range of a particular RSU.

- "Road-Side Host Activation": configuration files of the road-side services are sent to the DDS. From there they are globally available within CVIS. RS-Service → RS-FOAM (R-DDS) → RS-COMM → C-COMM → C-FOAM (C-DDS)

The user starts the vehicle

- "Vehicle Sub-system Start-up": the CVIS mobile host, router and gateway are started.
- name, password, smartcard or biometric data.
- "Vehicle Host Activation": [I am alive message] is send to the HMC; V-FOAM → V-COMM → C-COMM → C-FOAM

The driver can insert a destination (in that case a route is calculated)

- To be done with PTV and MIZAR

Configuration of CVIS vehicle host User → HMI → V-FOAM → V-COMM → C-COMM → C-FOAM

- "User Profile Activation": after authentication, the HMC will look up the stored end user profile (if it exists). Such a user profile, e.g. myCVIS, consists of all mandatory public services and optional free or paid services previously selected by the user.
- "Change User Profile": after user profile activation, the user will be given the opportunity to change his profile by adding or deleting (available) optional services from a list provided from the DDS. **Note** that it only involves the services which are available within a specified range of the vehicle, most likely the fuel range.
- "Synchronization": the HMC subscribes the user to the selected services and verifies, through the remote management protocol, whether the services belonging to the user profile (or the right version of them) are actually present on the CVIS vehicle host. If not, they will be provisioned by accessing the DDS including the properties (see 1) of the partnering road-side units.

  Note due to the so called "Selection Criteria" only the services which are available along the route (in case the route is known) or within a range of X kilometers.

The vehicle continuously monitors its location and the need to start communicating to a CVIS road-side host, e.g. the priority application.

If communication between the vehicle and road-side is set up, the vehicle sends the 'Minimum Data Set' to the road-side unit. Making itself known to the road-side unit represents the action 'Request for Green'.

- The minimum data set consists of: ID (to determine vehicle type), travel time to stopping line from current position (to determine time of arrival at stopping line), movement at the intersection (to determine the corresponding signal group) and status (heavy, dangerous, emergency, much too late, much too early, etc.).

The road-side unit collects all green requests, either cooperative or traditional via loop detectors, and answers the cooperative ones with a 'message received'. As far as possible all green requests are labelled with a certain priority level/weight, which are set by an authority.

The road-side unit calculates the optimal planning for the traffic light controller.

- The cost function of each link consists of variable and weights. One of the weights, most likely the weight for delays, represents the sum of the priority weights of all vehicles on a link. Additionally, congestion detectors are included in order to overrule the priority application during peak hours when the application is likely to perform poorly.

  Note: initially, all vehicles get 'conditional priority', which means priority when possible. Some vehicles, like emergency vehicles may require 'absolute priority' and therefore could be treated differently.

- The weight of the links only define which signal group will be realized first. When green should start and how long it should start depends on the queue, the capacity flow and the travel time to the stopping line of the priority requesting vehicle. Furthermore, a number of conditions apply:

  - All directions should have green at least once per cycle.
  - Conflicting signal groups may be cut off with preservation of minimum times.

    Note: it has to be considered if minimum green can be set dynamically.

- Facilitating priority request for conflicting directions requires a flexible traffic light control. With a traditional stage oriented control requires a certain cycle has to be gone through, which may result in a considerable delay before a specific signal group can be realized. To solve this problem a signal group oriented control is required, in which the control cycle is flexible and each signal group can be realized when needed.

The road-side unit sends the planning to the traffic light controller for implementation. Optionally, a return message may be send to the priority requesting vehicles. However, a number of remarks need to be made:

- The user acceptance of a system is very important. If a decision is made and the user in the vehicle is information about the current or future green window, it should not be able to change this anymore. Alternatively, a condition could be not to change a decision anymore once an informed vehicle has entered the dilemma zone and a hazardous situation may occur when the user abruptly

decides to brake thoroughly.

- Since the traffic situation at an intersection is very complex and highly dynamic it is very likely that there is a need to change the signal light planning on a very last notice. Therefore, it should be considered not to inform the user at all.
- An alternative for the above may be to allow changes in previous decisions and inform drivers about the reason for this change. This approach should only be applied in exceptional situations, like a last minute priority to an emergency vehicle which otherwise would have violated the red light. This example has a very strong relation with the SAFESPOT intersection application which objective is to warn road users at an intersection for red light violators.

- Once the vehicle has passed the intersection it will sign out. A *signing out* message enables to calculate driving times in the vicinity of the intersection easily.

7.7.5 High level composite architecture

This section provides specification of the high level composite architecture. The high level composite architecture describes the overall architecture of the system and the partitioning into sub-systems and components. It also identifies the interfaces and the relationships between the sub-systems, components and interfaces.

The high level composite architecture of the priority application is shown in Figure 169.

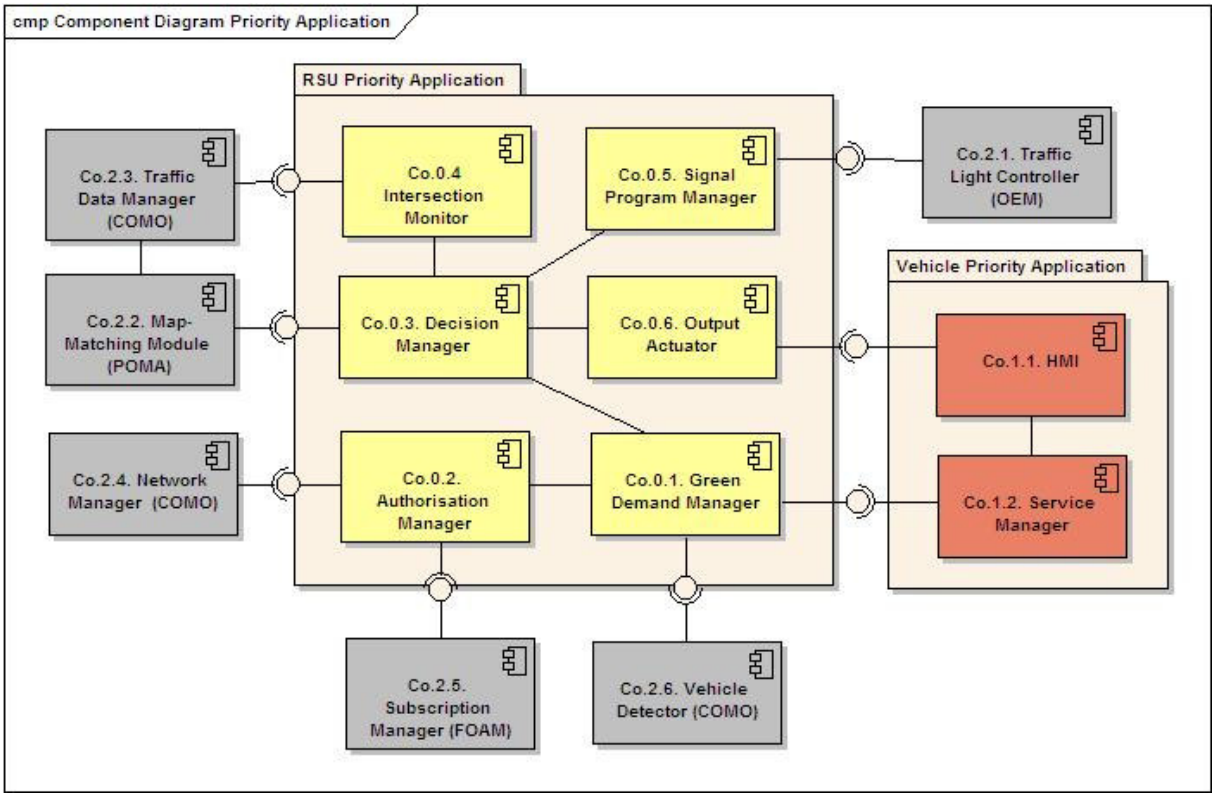

Figure 169: Component diagram priority application

The following main components are identified:

Green demand manager: responsible for collecting all green requests via both the traditional loop detectors and cooperative green requests from special category vehicles.

Authorisation manager: responsible for verifying the validity of the green requests and ranking of the request based on the priority levels

Decision manager: responsible for the handling of green requests and update of the signal program. Also initiator of return messages to the vehicle.

Intersection monitor: concerns an on-line monitoring function for the intersection. This involves: travel times, queue lengths, turning percentages, waiting times and traffic flows.

Signal program manager: responsible for extracting and implementing the signal program from/into the traffic light controller.

Output actuator: responsible the generation and distribution of return message to the vehicle.

HMI": responsible for presenting and receiving information to/from the driver of the vehicle.

Service manager: responsible for activating a service/application once it is available.

Traffic light controller: responsible for the implementing and monitoring of the signal program.

Map-matching module: responsible for mapping the information in order to facilitate the decision process (POMA).

Traffic data manager: responsible for the provision of specific traffic data like waiting times, queues, etc. at the intersection (COMO).

Network manager: responsible for the provision of traffic management strategies, in case priority levels in particular (COMO).

Subscription manager: responsible for the maintenance of the user profile where can be found what services each user/vehicle is subscribed to (FOAM).

Vehicle detector: responsible for vehicle detection and providing data to determine the traffic state at the intersection (COMO).

More details of these components can be found in the respective D.SP.3.2 specification documents.

7.7.6 Deployment model

This section describes the logical deployment of the priority application onto the CVIS infrastructure using UML deployment diagram as shown in Figure 170.

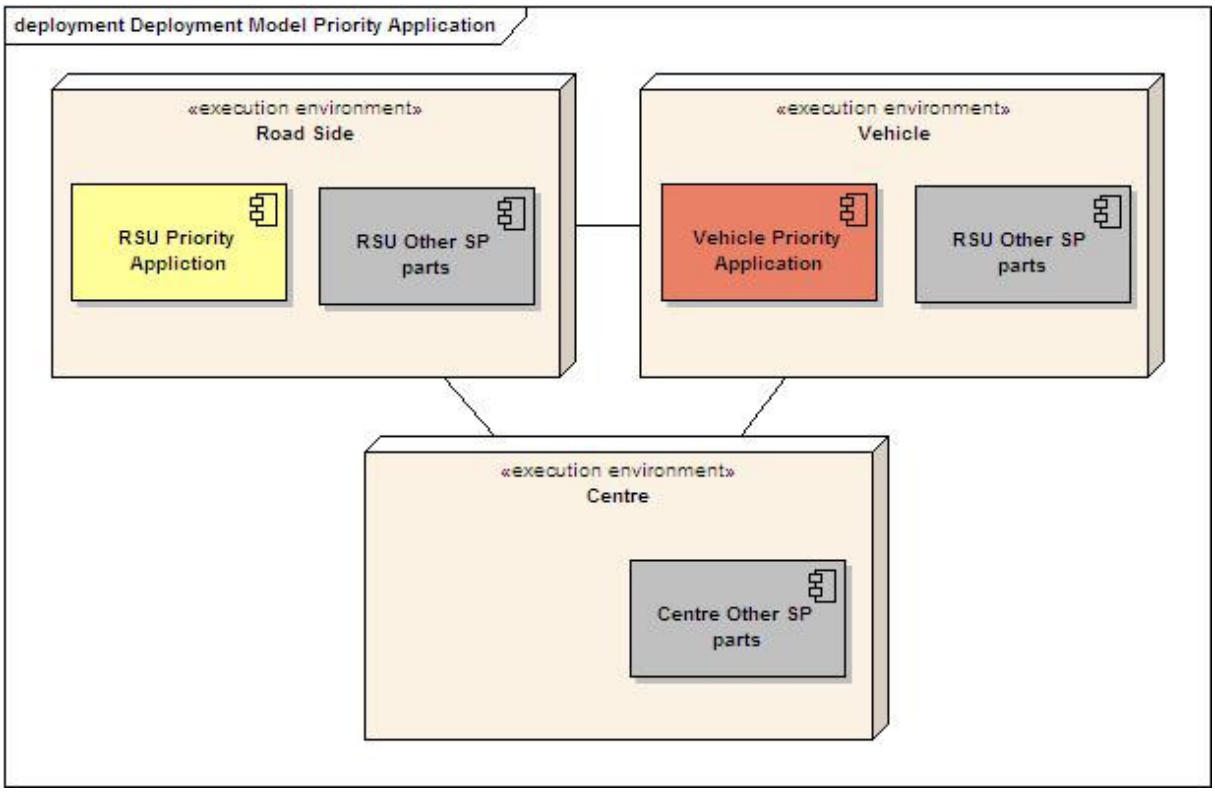

Figure 170: UML deployment diagram

7.8 Speed profile

The speed profile application and its main services are introduced in this sub-section. Further details of the speed profile application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.8.1 Overview

The speed profile application aims at providing speed advice to the driver in order to increase comfort and increase traffic safety. The application utilizes basic communication facilities and the location reference domain facility.

In the CVIS context, vehicles are equipped with an intelligent device that is able to communicate with the infrastructure and among vehicles. This intelligent system can thus help the driver to choose the best speed to approach signalised intersections.

The speed advice functionality would demonstrate that it is possible to communicate to the vehicles the best speed to approach the signalised intersection to have the green light. The speed profile is different for each direction and for different distance from the intersection.

The speed profile calculates the best approaching speed considering the remaining green time (or the time to green) and the strategy decides by the next signalised intersection. Then it

communicates to vehicles the best approaching speed profile, to maximize the intersection throughput (and also minimize travel time for the vehicles). It also calculates the remaining time of red time to alert the driver of the light changing, in case of red light.

The aim of the application is not only the optimisation of the drivers' comfort, but also the optimisation of the throughput of the intersection.

The speed advice functionality is studied for a series of intersection that have a "preferential" axe. In this way it is possible to communicate the vehicle not only the best speed for the next one intersection, but also for the next two (if the distance between the two intersections is not too high). It is not possible to calculate the speed to approach the third intersection, because the adaptive control changes control strategy each 3 seconds.

The system can be used in intersections which the distance from one between another is not more than 50 meters, but it is not very useful because the distance between the intersection correspond at about three seconds, and the traffic light phase can not be modified.

Main services and system boundary

This section provides an overview of the main services of the system and the system boundary. It also includes the mapping to FRAME artefacts (the traceability matrix) and description of non functional requirements (QoS).

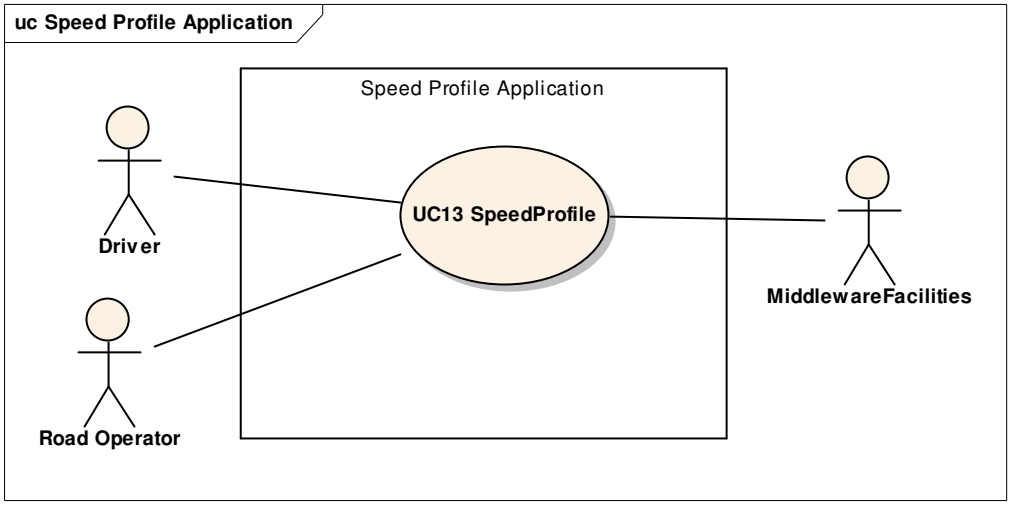

Figure 171: Use case model with system boundary

The use cases are as follows:

UC13: ***Speed profile:*** This use-case describes an important feature to increase efficiency of an intersection and the network. This is should be available on all CVIS enabled vehicles for maximum effect. The use-case has two main components. The first is to extend the controller algorithms to deal with more and more detailed information received from CVIS vehicles (FCD). The second main component is broadcast of speed profiles. CVIS technology is used to broadcast speed profiles to CVIS equipped vehicles driving in a certain direction at the intersection. The speed profile is a short time profile with speed advice (suggesting acceleration / deceleration or just suggesting the average approaching speed). The profile will be interpreted by the vehicles' on-board unit and if necessary presented to the driver. We also expect a positive effect on environmental issues due to the reduced amount of stops and go's.

The actors and their needs and responsibilities are described in the following:

Driver: The private driver wants to travel through the urban network in a comfortable, safe and efficient way. He is interested in being supported by dynamic information and navigation systems.

Road operator: Organisation responsible for maintaining the roads and managing the traffic on it. The road operator wants improve traffic control and traffic management by defining and implementing cooperative control and management strategies.

Middleware facilities: Represents the CVIS basic and domain facilities (see part II of this document)

For an application the main external interface is the interface provided to the end user. (Note that we are not concerned about the GUI at this point, but the description of the services provided to the end user). The application programming interface for the speed profile application is shown in Figure 172.

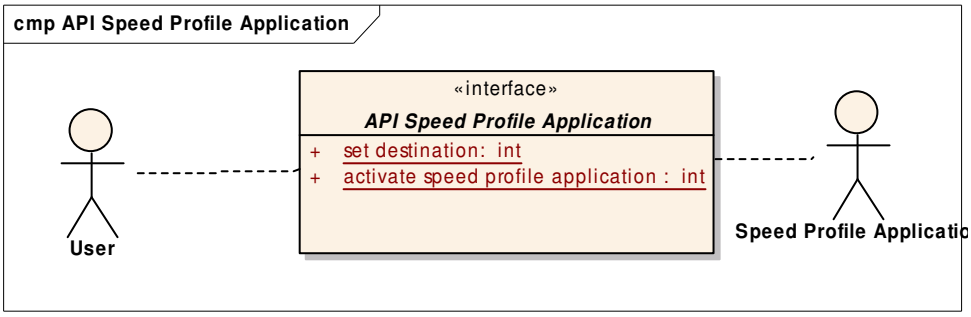

Figure 172: API speed profile

The behavioural aspect of the interface is shown in Figure 173.

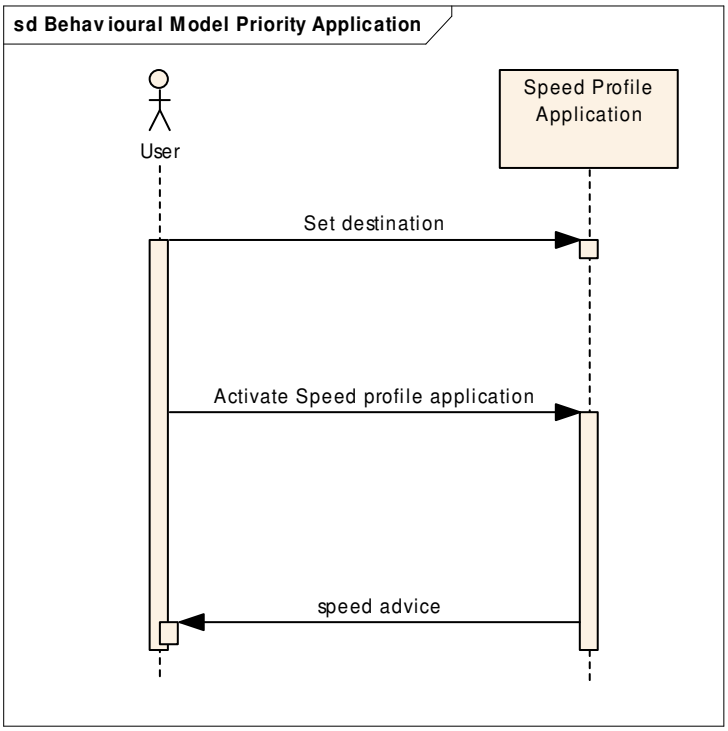

Figure 173: Behavioural model speed profile

7.8.2 Information model

This section provides the specification of the information model. The information model identifies and defines the main concepts of the application domain. The concepts are specified in terms of their types using UML class diagram as shown in Figure 174.

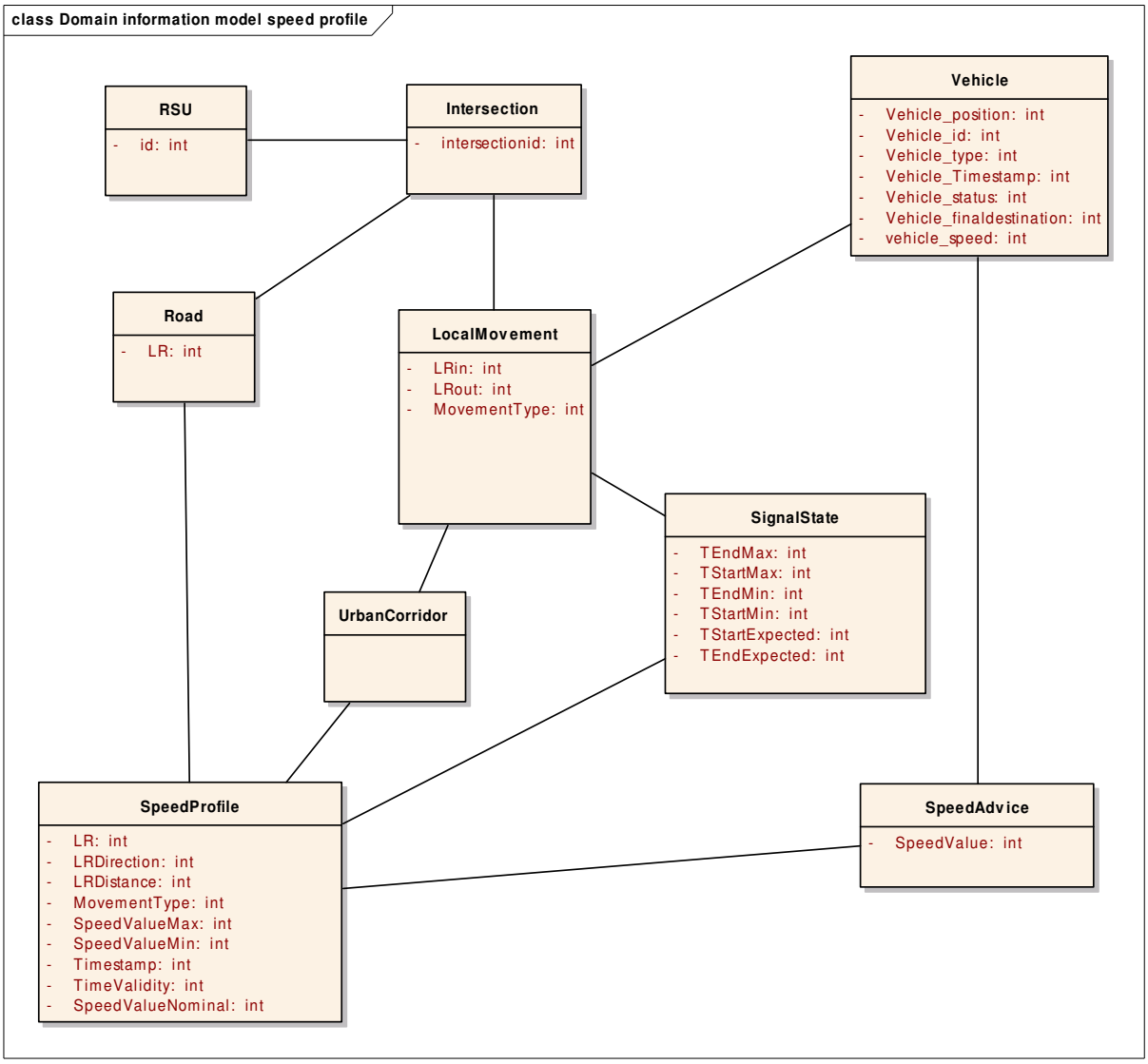

Figure 174: Domain information model speed profile

7.8.3 Interaction model

This section provides specifications of main domain processes. The overall value chain of the priority application is shown in Figure 175.

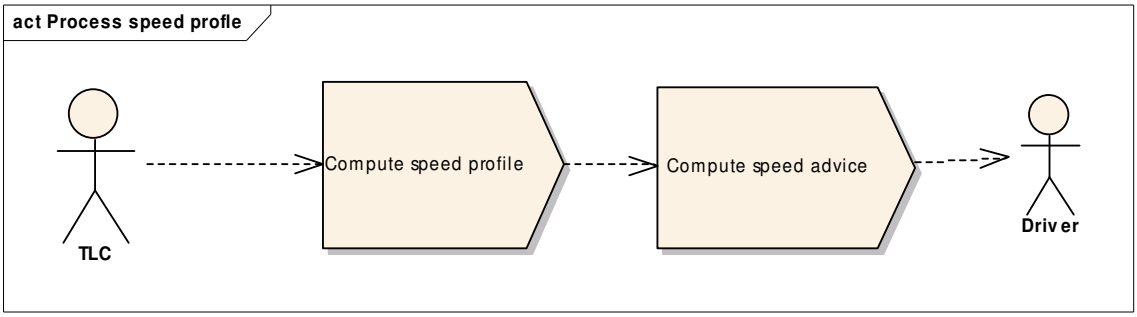

Figure 175: Reference service process speed profile

The workflow of the priority application is specified using UML activity diagram as shown in Figure 168.

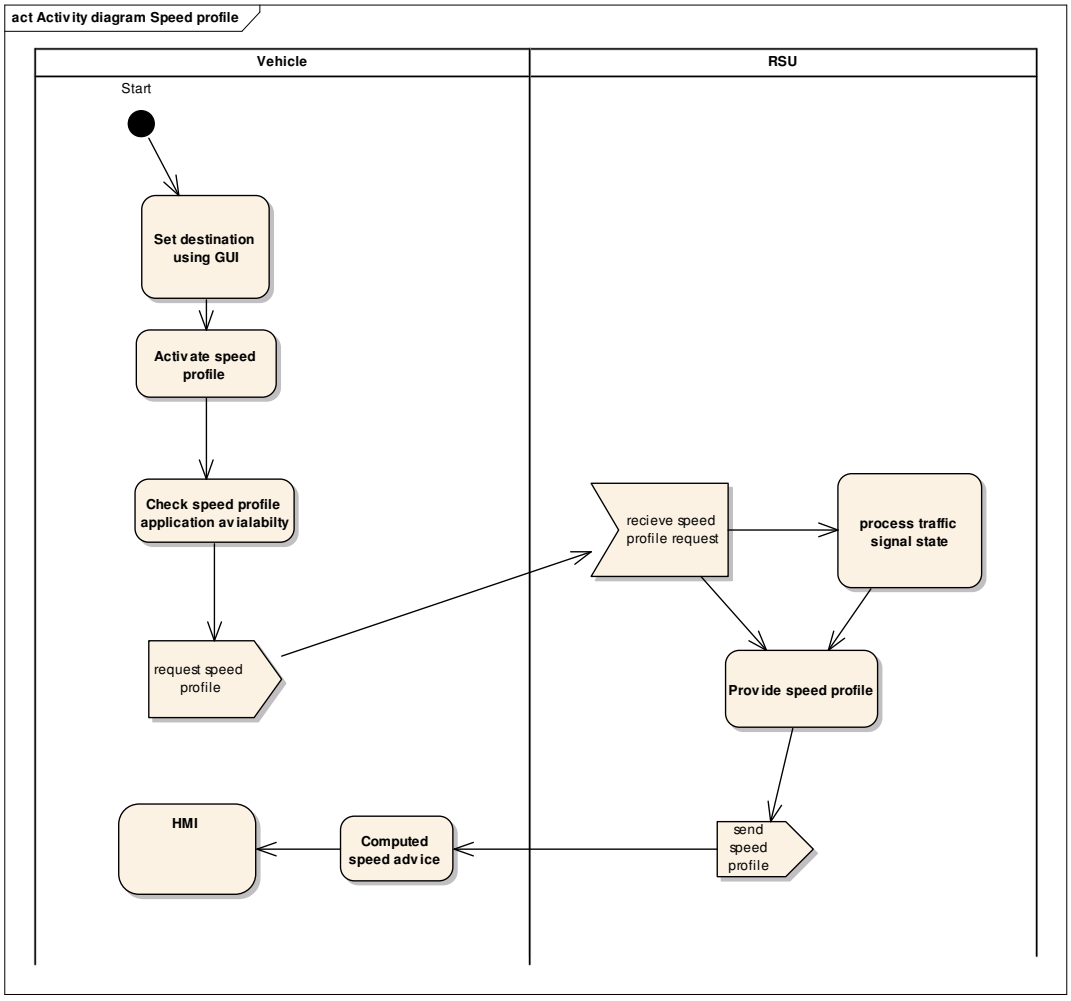

Figure 176: Activity diagram speed profile

Scenario description

1. The vehicle knows type of vehicle, actual speed, location and direction of the CVIS vehicles. In this way the situation on the road is known in the most detailed way.
2. Controller calculates new speed profiles for every direction (time horizon of the profile might differ for different directions).
3. Controller broadcasts those profiles to all equipped downstream vehicles in one direction.
4. CVIS equipped vehicles receive the message and compute the speed advice suitable for the vehicle state.
5. The driver is notified in a suitable way, e.g. blue arrow up to speed up and red arrow down to slow down a bit for a more fluid ride, arrows might be dynamic in size.

7.8.4 High level composite architecture

This section provides specifications of the high level composite architecture. The high level composite architecture describes the overall architecture of the system and the partitioning into sub-systems and components. It also identifies the interfaces and the relationships between the sub-systems, components and interfaces.

UML composite structure diagrams, UML class diagram and/or UML component diagram can be used for the modelling. It is recommended to use the UML composite diagram, which support the necessary mechanisms to model several levels of abstraction (composites) in one diagram in an intuitive way. UML composite structure diagram was introduced with the UML version 2.

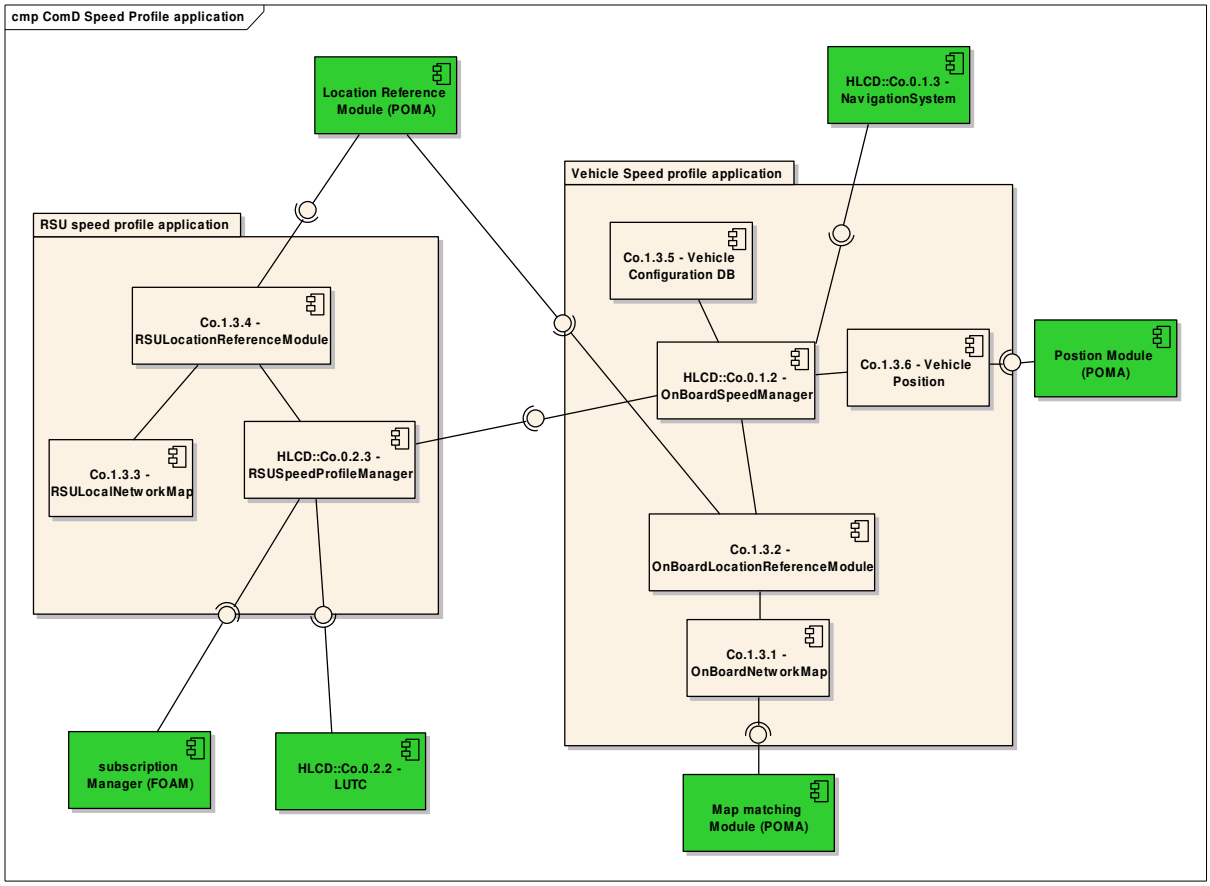

Figure 177: Component diagram priority application

The following main components are identified:

OnBoardNetworkMap": The on-board network map is used to filter the messages coming from the RSU and to map-match the current vehicle position.

LocationReferenceModule": The location reference module on the vehicle is used to convert the information coming from the infrastructure. The information is used to filter the relevant information for the vehicle, matching the current position and the received one on the local map (POMA)

Vehicle position: The component provides current and possible next vehicle position.

Vehicle configuration DB: The component manages the access to the configuration DB of the vehicle, this DB contains information on the vehicle type, mass, acceleration and deceleration ranges, status, category, e.g. commercial vehicle, public transport, emergency.

RSULocalNetworkMap": The onboard local map has the information used to represent using a location reference technology the speed profile information to be communicated to the vehicles.Local map has also the matrix of movements.

RSULocationReferenceModule": The RSU location reference Module is responsible to translate back/forth the location information to the local map.

Subscription manager: responsible for the maintenance of the user profile where can be found what services each user/vehicle is subscribed to FOAM.

Position module: Provides the position of the vehicle

Map-matching module: responsible for mapping the position information in order to facilitate the decision process.

7.8.5 Deployment model

This section describes the logical deployment onto the CVIS infrastructure, i.e. the deployment of the above specified components to physical nodes such as vehicles, road-side unites, management centres etc. UML deployment diagrams are used for the modelling.

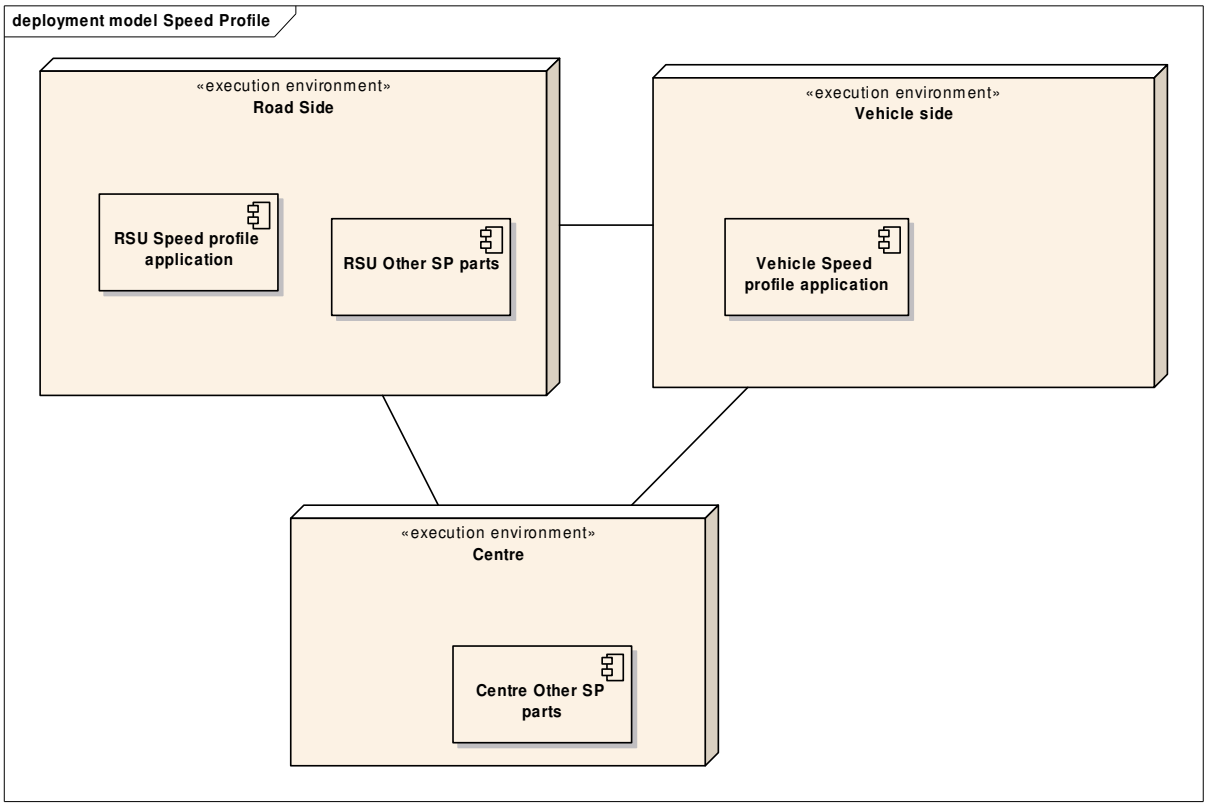

Figure 178: UML deployment diagram

7.9 Cooperative traffic control

The cooperative traffic control application and its main services are introduced in this sub-section. Further details of the cooperative traffic control application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.9.1 Overview

The cooperative local traffic control optimizes traffic flows in a limited control area, based on all kind of traffic data available. "Floating Car Data" (FCD) are collected and merged with data from traditional detection. This leads to an improved traffic information and is the basis for calculating the local traffic status for selected urban areas. In accordance with the traffic strategies defined by the local authorities, a master unit co-ordinates traffic flows of the subordinate intersections. This leads to an improvement of the overall network efficiency.

The activity flow roughly works in the following way:

CVIS vehicles in a defined control area send out FCD to the closest road-side unit. The road-side unit collects the FCD and merges them with available detector information. On the basis of both sources of information, the road-side unit sends local traffic states to the area master unit which coordinates a defined number of intersections. With all information available, the area master unit analyzes the traffic situation in the control area and sends out control measures to the RSUs as indicated in Figure 179.

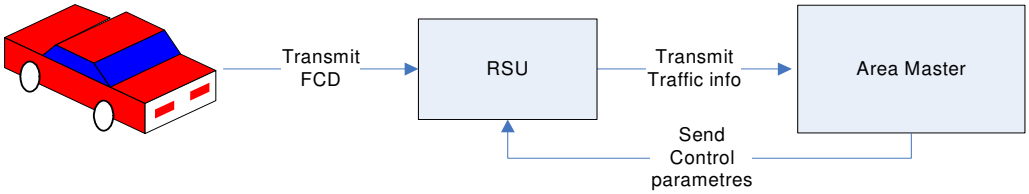

Figure 179: Cooperative traffic control overview

Main use cases and system boundary

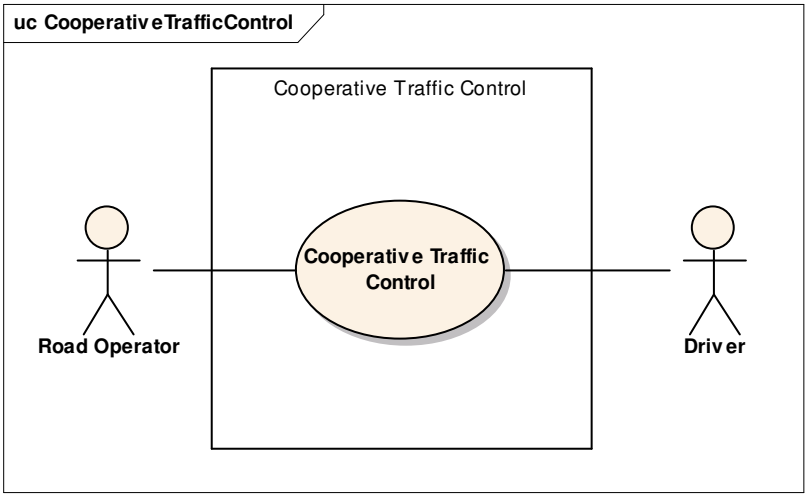

Figure 180: Use case model with system boundary

The use cases are as follows:

Cooperative traffic control: The goal of this use case is to optimize traffic flows in a limited area (up to 5 intersections), based on all kind of available traffic information (esp. XFCD and road-side sensor data) and using different methods of traffic control. Within the control area, one co-operative intersection will be equipped with a Master device; the others will be treated as sub-ordinate nodes.

The actors and their needs and responsibilities are described in the following:

The private driver wants to travel through the urban network in a comfortable, safe and efficient way. He is interested in being supported by dynamic information and navigation systems.

Road operator: Organisation responsible for maintaining the roads and managing the traffic on it. The road operator wants improve traffic control and traffic management by defining and implementing cooperative control and management strategies.

7.9.2 Application programming interface

The cooperative traffic control API is specified in Figure 181.

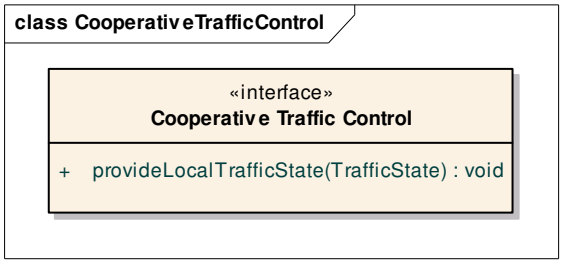

Figure 181: Cooperative traffic control API

The cooperative local traffic control does not require an interface with the end user, understanding that the driver is considered the end user. It is assumed that the FCD (time-stamped location of the vehicle) is automatically transferred to the next RSU without any active input from the driver.

This is elaborated in the sequence diagram below.

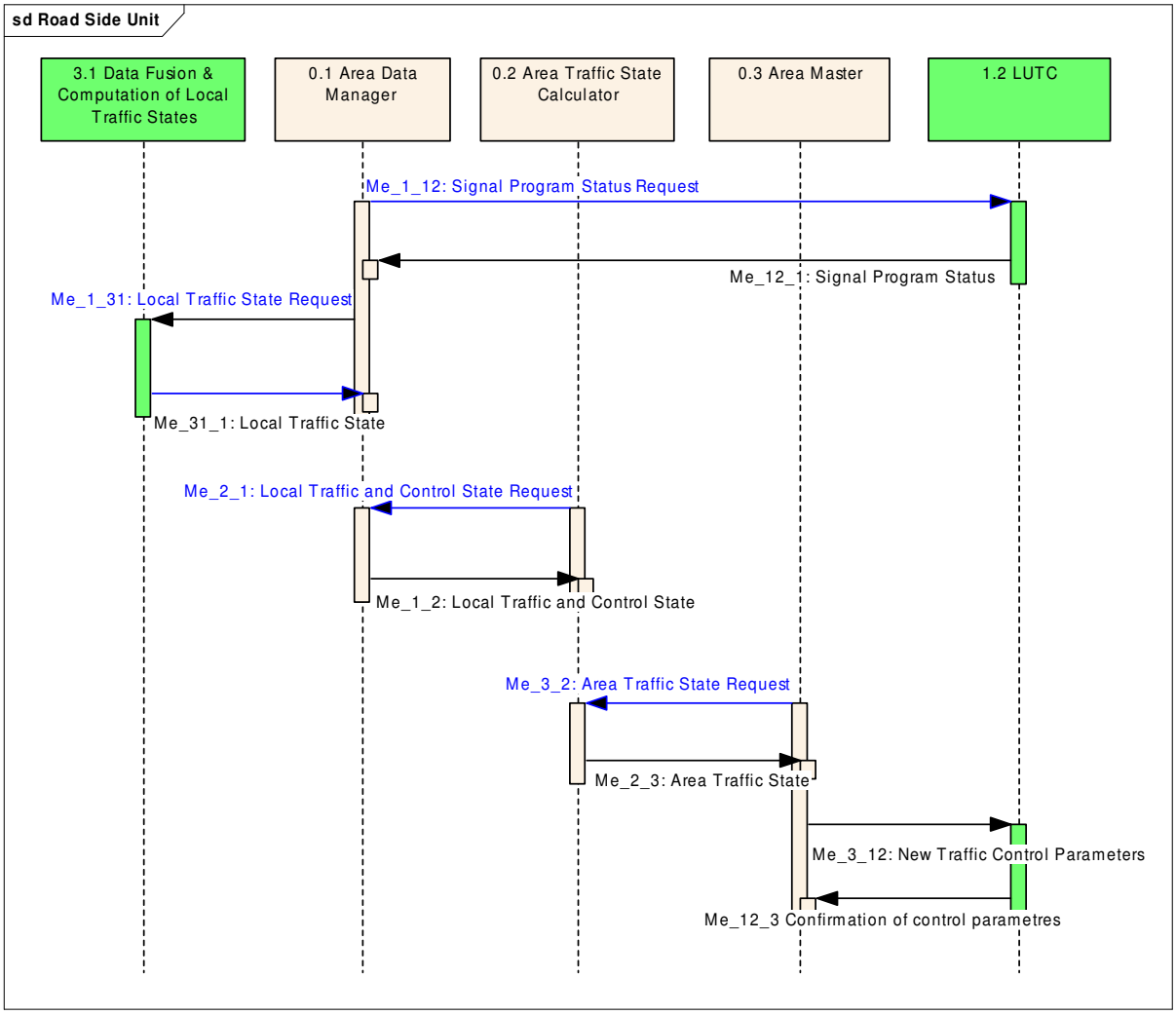

Figure 182: Sequence diagram cooperative traffic control application

7.9.3 Information model

The information model related to the cooperative traffic control application is shown in Figure 183.

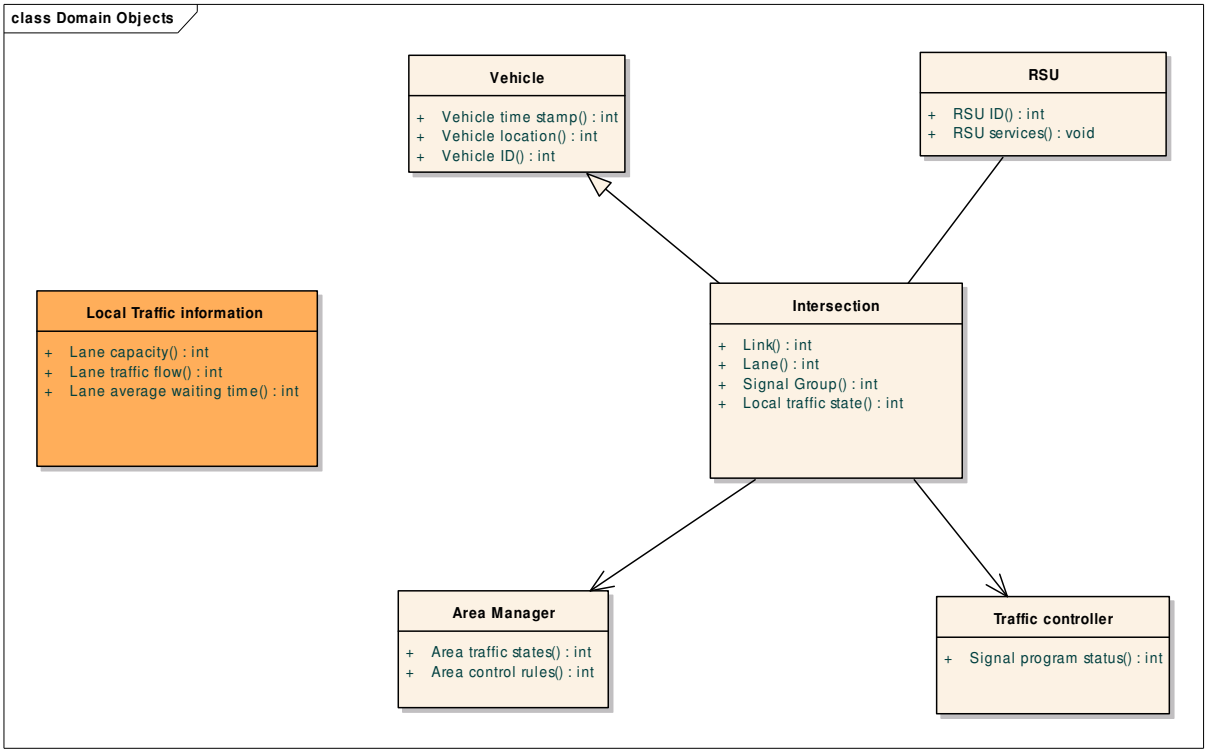

Figure 183: Domain information model cooperative traffic control application

7.9.4 Interaction model

The sequence in the cooperative traffic control is as follows:

XFCD is transmitted from all CVIS vehicles in the area to the closest road-side units (RSU) in real time.

The RSUs of the co-operative intersections receive the XFCD, perform a data fusion with the available detector data and compute the local traffic state for each intersection area.

The local traffic states are transmitted to the co-operative master intersection controller and are aggregated, resulting in a complete traffic state for the whole area.

On the basis of the area traffic status the co-operative master controller determines the optimal control measures for the whole area.

(Optionally, the information on the area situation is transmitted to the central system at network level.)

(Optionally, the central system checks the area control measures with regard to compliance with overall network priorities. Only in case of conflict, the central system sends a correction message back to the master controller.)

The traffic control strategies are transmitted to the subordinated co-operative intersection controllers.

The subordinated controllers implement the traffic control measures accordingly.

The overall workflow is indicated in Figure 184. The green processes are external processes (external to the cooperative traffic control application.

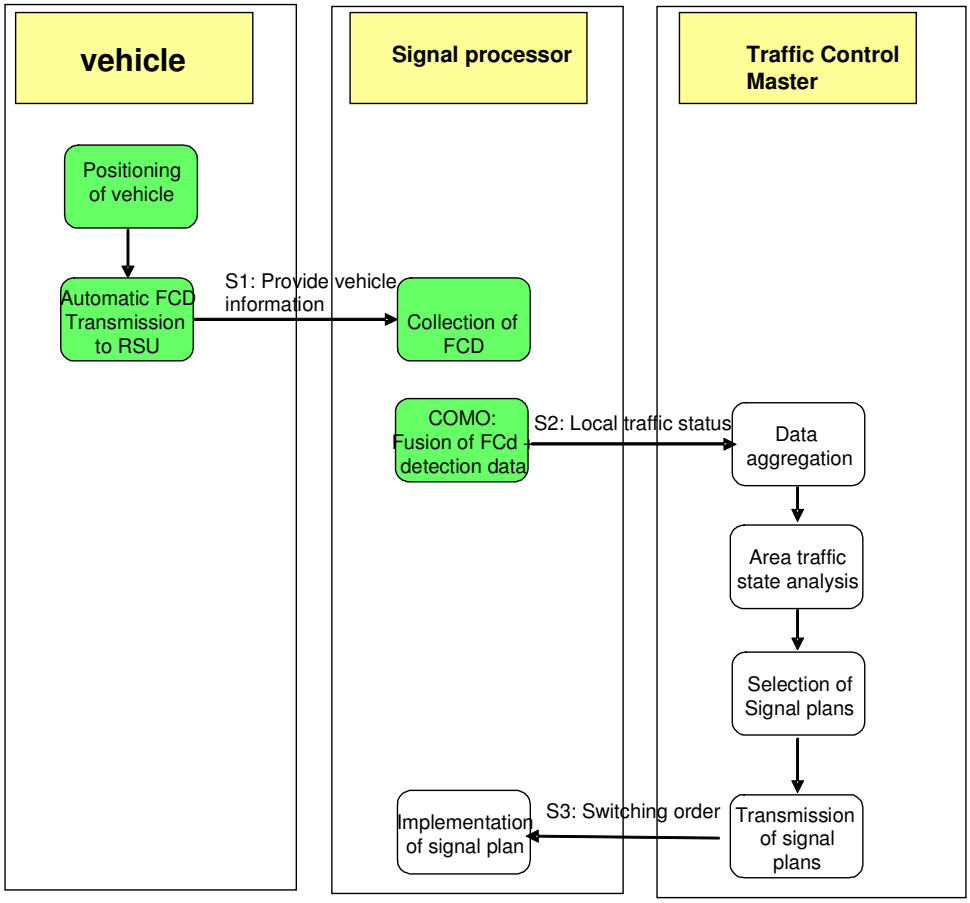

Figure 184: Overall workflow of the cooperative traffic control application

7.9.5 High level composite architecture

The high level composite architecture of the cooperative traffic control application is specified in Figure 185. The green components are external components (external to the cooperative traffic control application).

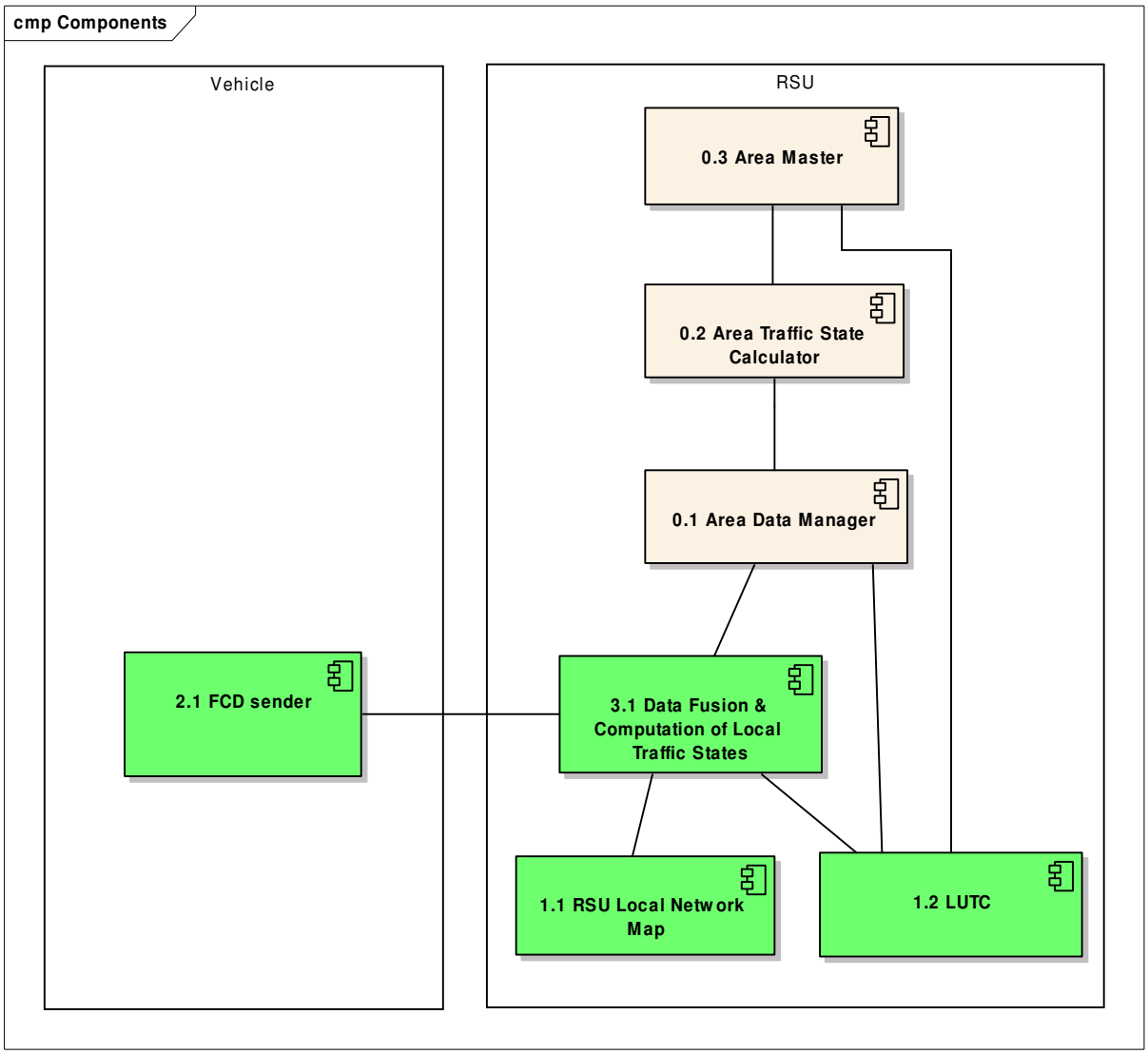

Figure 185: High level composite architecture, cooperative traffic control

The components are elaborated further below

Area master receives the area traffic state from the area traffic state calculator. It also receives the current signal programme status from the LUTCs in the control area. Based on this information, it algorithmically determines control parameters for the traffic light control of the area, aiming at optimizing the relevant traffic flows in the area. Finally the area master sends the traffic light control parameters down to the LUTCs.

Area traffic state calculator receives local traffic and control states from the area data manager. It checks the plausibility of the data, performs certain aggregation, and possibly selects relevant data subsets for further consideration. Eventually, it combines the local traffic states to one area traffic state.

The **area data manager** is responsible for bringing together all relevant data needed for the area traffic state determination. This data is sent to the area traffic state calculator of the master RSU. In case of a slave RSU this means that the data has to be sent from one RSU to another (neighboured) one.

Data fusion & computation of local traffic states: The data fusion manager receives FCD from the vehicle, formats them and pre-processes them. In parallel, it receives detection data from the LUTC. Pre-processed FCD and detection data are merged and used for the calculation of the local traffic state.

FCD sender is located in the vehicle and transmits FCD to the next RSU.

The **local urban traffic control** (LUTC) has the control methods for the conventional traffic control.

The **RSU local network map** contains information on the positioning of the CVIS vehicles in the intersection area.

7.9.6 Deployment model

The deployment of the cooperative traffic control application is according to Figure 185.

7.10 Flexible bus lane

The "Flexible Bus Lane" application and its main services are introduced in this sub-section. Further details of the flexible bus lane application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.10.1 Overview

The flexible bus lane application allows a driver within its private vehicle to access a reserved bus lane (BL) using available traffic data, route guidance, PT ETA and sends licenses to equipped vehicles. All equipped vehicle may contribute to the infrastructure, furnishing traffic data, queue at traffic lights, and estimation of vehicle flow, incidents/accidents, and many other useful information for the traffic management system.

A typical scenario of use is contextualized in the urban environment. A driver of a private, CVIS-equipped vehicle should use a route guidance system to reach his destination. During the computation of the routing, "Bus Lanes" (BLs) may be considered as alternatives. In this case, when a vehicle is approaching a BL that may shorten his routing, a prompt may ask the driver to use the BL (as alternatives, driver may know if the BL shorten his journey). This question must be raised only if the AVM system in collaboration with the RSU estimates no queue at the next stop/crossing/traffic lights for "Public Transport" (PT) vehicles. If the driver accepts, a license for accessing the BL will be given to on board CVIS-equipment. When a private vehicle approaches a BL, it can go through it, but it can be asked to leave the BL in case of approaching PT vehicles or in case the ETA of following PT vehicles is not kept. At every crossing/traffic light the driver/private vehicle must request a license renewal. If ETA, traffic congestion, flow, accident/incident and other information gathered by RSU allow license renewal, the driver may continue to use the BL, otherwise the BL must be left. To enforce the received instructions, some enforcement may be installed, e.g. video enforcement.

The flexible bus lane application is depicted in Figure 186.

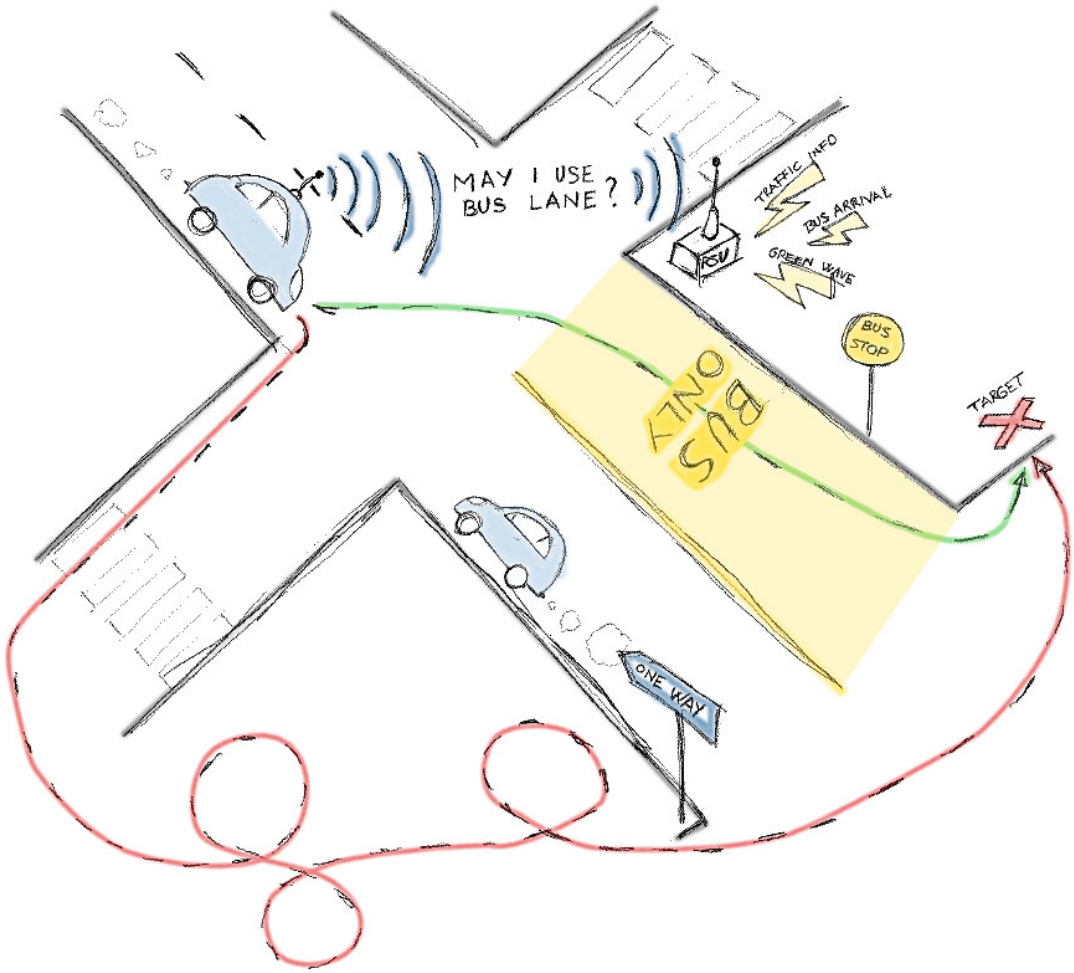

Figure 186: Non-formal representation of the flexible bus lane application

Main use cases and system boundary

The main use cases and the system boundary are specified in Figure 187. The flexible bus lane application use cases can be associated with routing use cases to provide more advanced urban traffic management. This is also indicated in Figure 187.

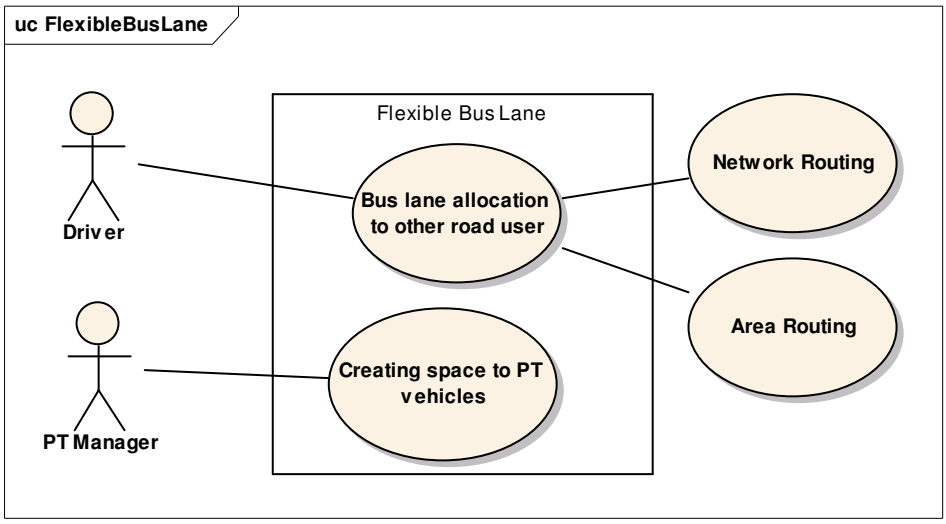

Figure 187: Use case model with system boundary

The use cases are as follows:

Bus lane allocation to other road user: The goal of this use case is to increase the capacity on dedicated road sections by providing BL access to CVIS equipped road users when the BL is not used by public transport or other specific vehicles.

Creating space to PT vehicles: The goal of this use case is to clear the BL in order to avoid disturbances for public transport and to guarantee a punctual and fast transit for the public transport vehicles.

Request for green: Creation of a "requested green" in a cooperative way (targeted at special road user categories). This use case describes the creation of a, vehicle requested, green on a controlled intersection. This requested green is used to give priority to special road categories like emergency vehicles, trucks with DG, public transport etc. For those categories, it aims at a more fluid (and safe) intersection crossing. Priorities for the different categories could and should be set by authorities. A feedback loop to the drivers is optionally available.

Area routing: The goal of this use case is to provide route guidance advice generation at local and distributed level. The area of influence can span from a single RSU, to a cluster of RSUs. The cluster can depend on the network and the actual position of the constraint. The information can then be propagated at higher level, to generate network level route guidance. The system will consider an incident or momentary disturbance in traffic flow. By conducting the determination of incidents and the calculation of alternative routes within the local road-side infrastructure a highly responsive local area rerouting is realised. The private vehicle drivers benefit from a reduction in travel time and an increase of safety, while preserving network efficiency.

Network routing: The goal of this use case is to support the implementation of network wide traffic scenarios by giving routing advice to individual vehicles while knowing their

characteristics and destination and also the network wide traffic conditions. By using individual information the road operator is able to offer, via a service provider, several individual route options that cannot be offered in a collective way, e.g. by road-side displays. The road operator can achieve better distribution of traffic by routing vehicles via links with capacity resources.

The actors and their needs and responsibilities are described in the following:

The private driver wants to travel through the urban network in a comfortable, safe and efficient way. He is interested in being supported by dynamic information and navigation systems.

Road operator: Organisation responsible for maintaining the roads and managing the traffic on it. The road operator wants improve traffic control and traffic management by defining and implementing cooperative control and management strategies.

Public transport operator: The public transport operator is a private or public commercial entity providing and managing public transport in the urban network. The public transport operator wants his vehicles to travel in a fast, safe and efficient way.

The API of the flexible bus lane application is specified in Figure 188.

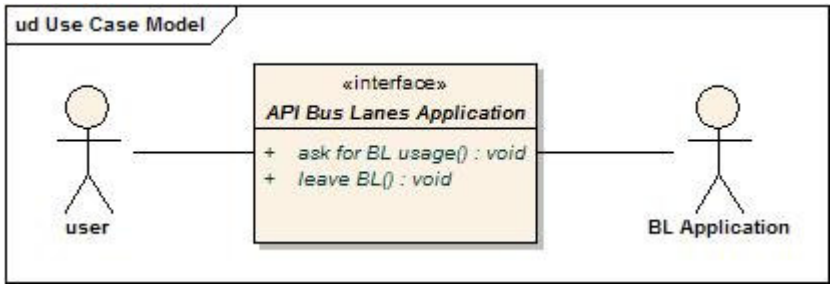

Figure 188: API flexible bus lane application

The methods defined through the interface described are:

ask for BL usage, from the user to the application;

leave BL, from the application to the user

An example scenario applying these methods is specified in Figure 189.

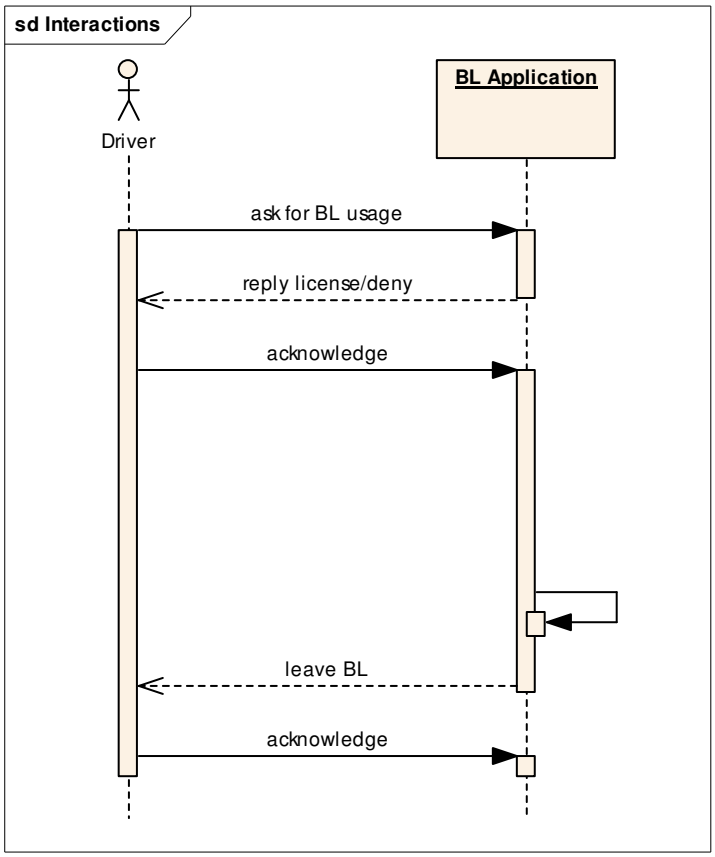

Figure 189: Behavioural model flexible bus lane application

7.10.2 Information model

The domain information model of the flexible bus lane application is specified in Figure 190.

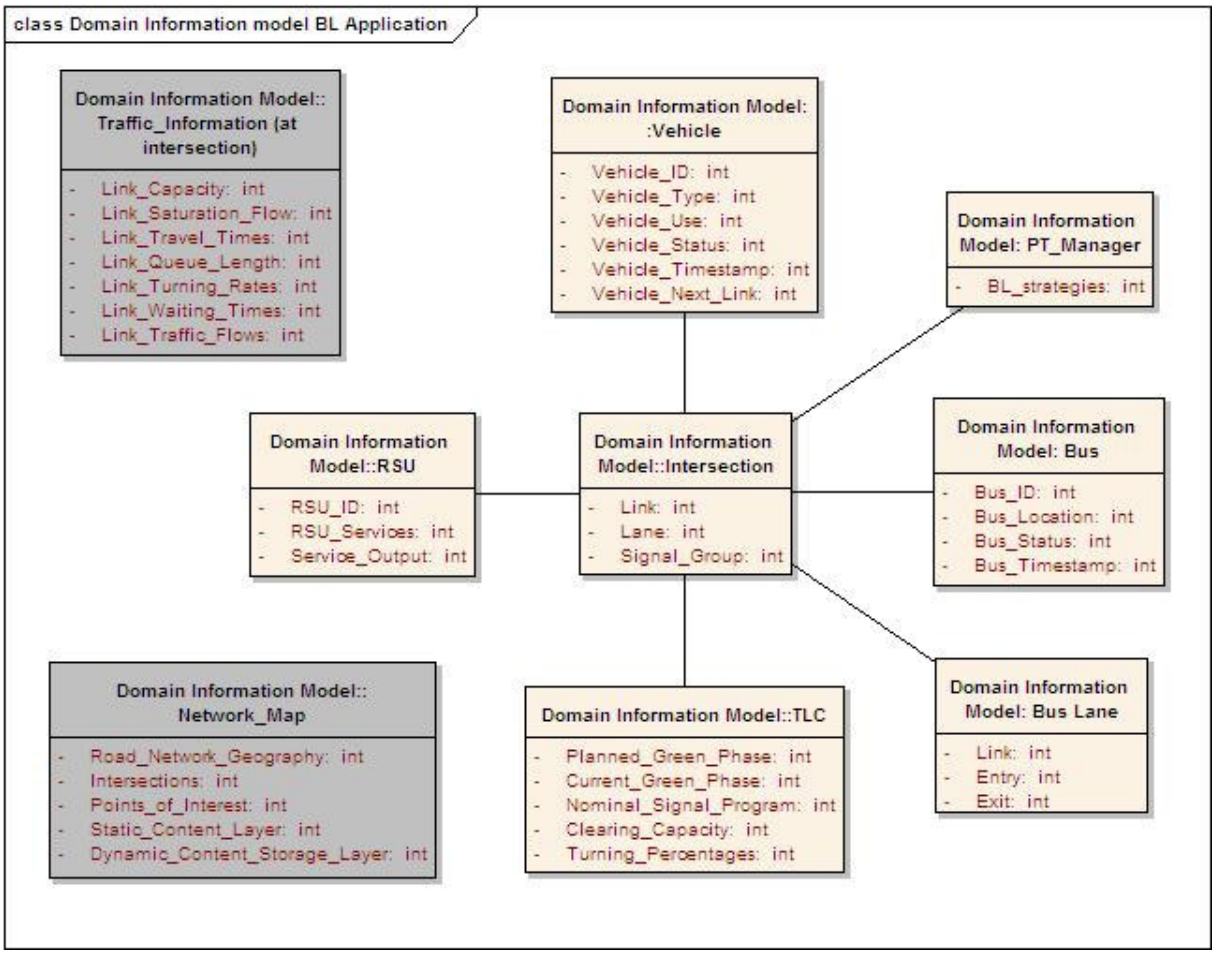

Figure 190: Domain information model flexible bus lane application

7.10.3 Interaction model

The flexible bus lane application comprises two main use case scenarios: 'Bus lane allocation to other road user' which provides licenses to use the BL and 'Creating space to PT vehicles' (preventing disturbances to public transport vehicles). In detail the sequence of events of the flexible bus lane application for these two use cases is as follows.

Bus lane allocation to other road user

1. Public transport vehicles send ID and position to the PT management centre.
2. PT management centre sends to the RSU data related to the buses arriving to the BL section in the next 15 minutes (ID, destination, ETA, etc.).
3. CVIS equipped vehicle approaches and reports its destination, vehicle characteristics and speed to the CVIS road-side controller.

4. In case of congestion or expected congestion on the following road section a license to use the BL is requested from the vehicle to the RSU of the current section. If the licensed CVIS vehicle is already using the BL and/or the time of expiration of the license is over, it has to request the renewal of its license to the following RSU when changing section or current RSU when still into the section.
5. CVIS road-side controller decides on basis of CVIS vehicle characteristics and speed, intended destination, global traffic flow, next bus ETA, bus stops and dwelling times along the BL, BL occupancy, queue at crossing / traffic lights and headways needed for public transport, if CVIS vehicle approaching is allowed to use the BL. The aim is to optimise general traffic situation without jeopardising the level of service of PT e.g. without increasing PT vehicles waiting time at traffic lights.
6. If the approaching CVIS vehicle does not create disturbances, RSU releases a new license to the CVIS vehicle in order to use the BL. The license is communicated to the driver via HMI and is valid for the estimated duration of completion of the section by the CVIS vehicle.
7. RSU gets back an acknowledgement message from the driver.
8. If the CVIS road-side controller decides not to release a licence, RSU stops to release new licenses to CVIS vehicles until the conditions for granting a license are fulfilled again.
9. RSU restarts to release new licenses after the bus has entered in the BL (therefore coming back to point 6).

Creating space to PT vehicles

1. Public transport (PT) vehicles send ID and position to the PT management centre.
2. PT management centre sends to the RSU data related to the next bus approaching to the BL (esp. ETA at the traffic lights).
3. CVIS road-side controller analyses the traffic situation on its BL section and determines the waiting time at the next traffic light for public transport vehicles.
4. CVIS road-side controller identifies the private vehicles running in the BL which distance to the approaching bus is smaller than the minimum headway needed to avoid queue at the traffic light.
5. If the headway is inadequate and/or a green wave not feasible, no further licences are granted until the bus has completed the BL section and the conditions for granting a license are fulfilled again.
6. CVIS road-side controller identifies the private vehicles running in the BL which license is no longer valid, e.g. time of expiration is over.
7. In both cases (points 4 and 6), RSU orders CVIS vehicles to leave the BL at end of its section or next crossing / traffic light (depending on the layout).

8. If the situation is "strongly critical", e.g. minimum headway not possible, green wave not feasible, heavy congestion on regular lanes and at next crossing / traffic light, serious incident / accident forbidding vehicles to be absorbed from the BL…, road-side controllers can communicate to the following controllers, access restriction to private CVIS vehicles and pre-emption procedure in order to prevent BL congestion and/or queuing at crossing / traffic lights.
9. The plates of non CVIS vehicles and CVIS vehicles without license are captured via video enforcement system and RSU respectively.
10. RSU sends plates and other information (time, vehicle category, location… when available - indeed for non CVIS vehicles it might be more difficult to get accurate time and coordinates especially if no video enforcement system is available) to the violation management centre.
11. RSU collects all traffic information about both BL and normal lane along its section (number of licenses, queue length, speeds, violation details, statistical data…).

A specification of the BL allocation is shown in Figure 191.

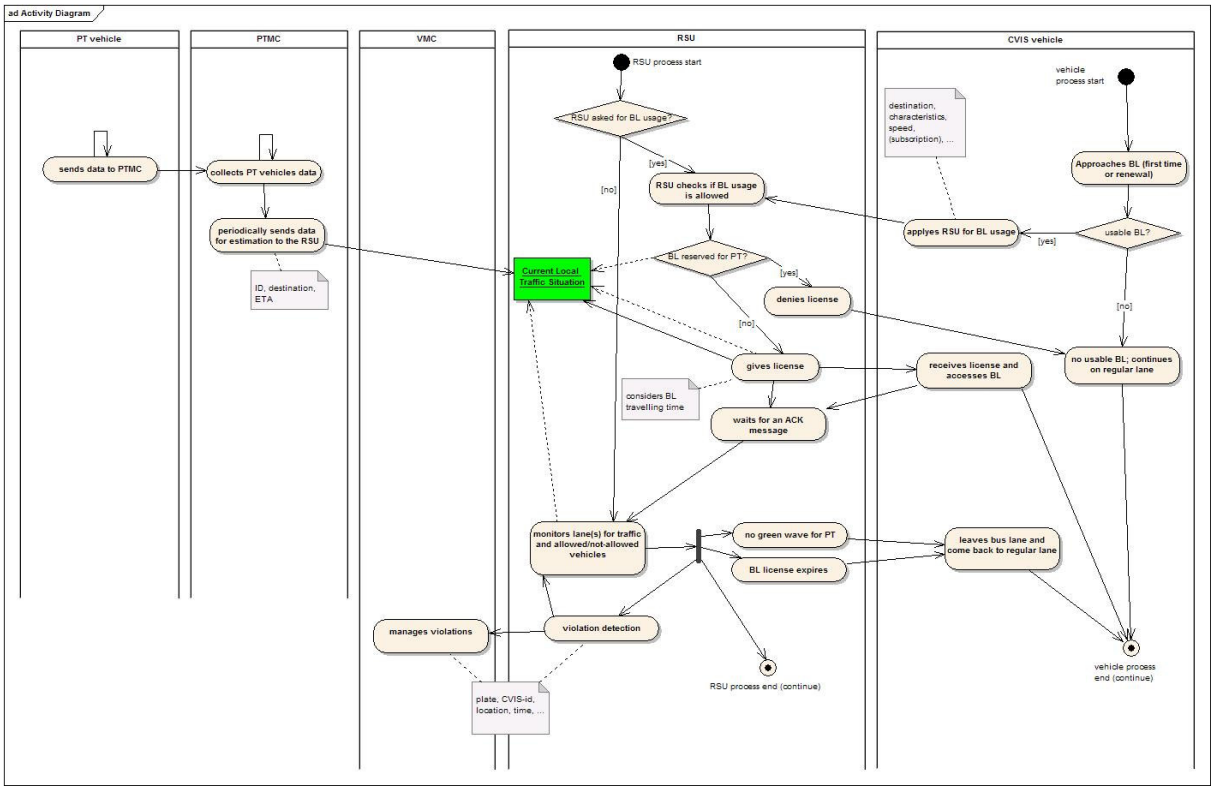

Figure 191: Activity diagram BL allocation

7.10.4 High level composite architecture

The high level composite architecture model of the flexible bus lane application is specified in Figure 192.

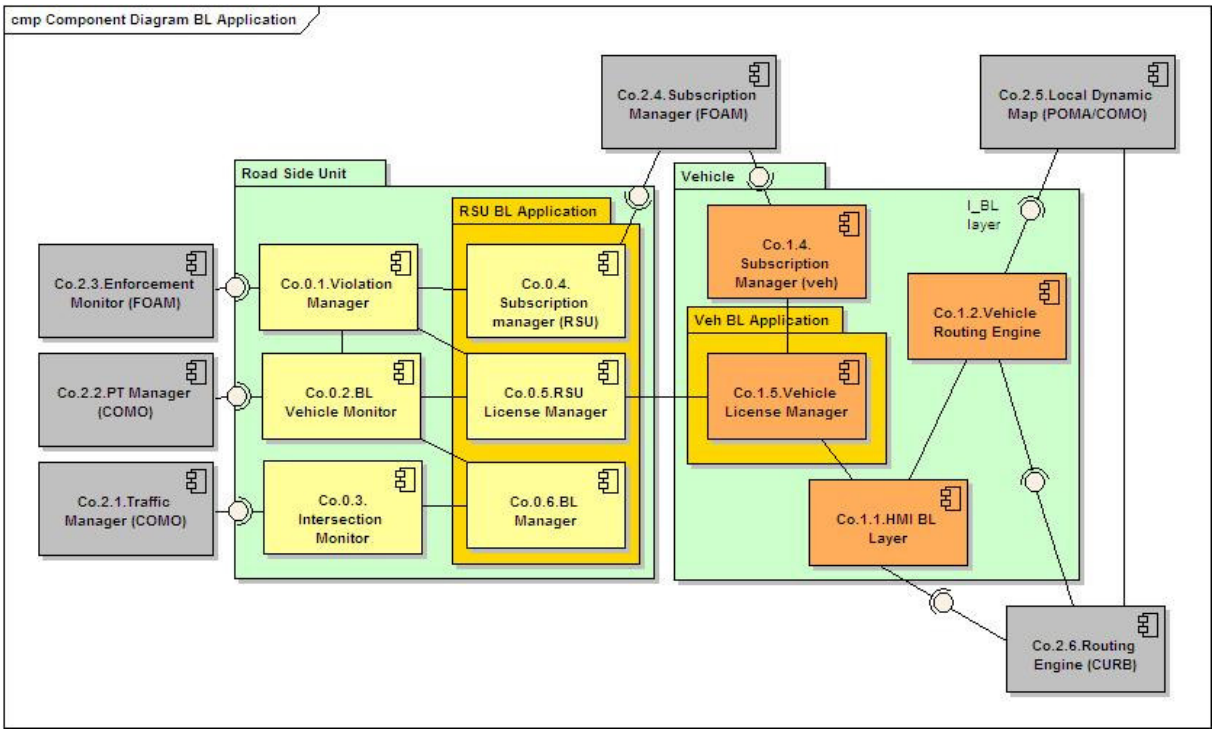

Figure 192: Component diagram flexible bus lane application

Co.0.1.Violation manager: manages any kind of violation forwarding to each suitable manager the actions-set to perform. It transmits any collected data regarding the detected violation.

Co.0.2.BL vehicle monitor: monitors the vehicle when it's allowed to access a BL, from its entry to its exit. It intercepts any information regarding PT vehicles (data coming from PTMC) to empty the BL and/or to deny access to further vehicles.

Co.0.3.Intersection monitor: It's the component inside the RSU that gives the estimated time to reach a green wave/go through a road section (both BL and not).

Co.0.4.Subscription manager (RSU): check that a subscription sent from a vehicle to use a service is still valid.

Co.0.5.RSU license manager: maintain licenses distributed to the allowed vehicles to access a given available service.

Co.0.6.BL manager: It's an RSU BL application component that manages the requests for the usage of a BL. It manages both applied licenses and check consistency with subscriptions before to release any license. It checks BL needs to be emptied computing data gathered by BL vehicle monitor.

Co.1.1.HMI BL layer: It's an extension of the usual HMI interface to allow the usage of the BL application. Through that it's possible to ask the RSU to use a BL, to select a start/end point for a journey, to reply at every message sent by BL application.

Co.1.2.Vehicle routing engine: It's a component that extends the routine engine (CURB) that builds journeys considering BLs usage (BLs may be seen as POI - point of interest - to be reached during journey).

Co.1.4.Subscription manager (veh): maintain any subscription that a vehicle (or driver) owns to access a set of services.

Co.1.5.Vehicle license manager: maintain licenses to certificate the lawfulness of a service use.

Co.2.1.Traffic manager (COMO): gives information to allow a RSU to calculate the queue at next cross/traffic light, to go through a road section, to go through a BL.

Co.2.2.PT manager (COMO): It gives information regarding the ETA, actual positions and delays/advances of PT vehicles.

Co.2.3.Enforcement monitor (FOAM): manages any information regarding a vehicle violation, and redirect to the appropriated institution.

Co.2.4.Subscription manager (FOAM): allow subscriptions interfacing between subscription managers (RSU and vehicle).

Co.2.5.Local dynamic map (POMA/COMO): It represents the "cartography" and services that allow the localization of the vehicle inside a given region. It also provides enriched maps that may be swapped following the application to use.

Co.2.6.Routing engine (CURB): It gives a journey starting from the initial point and ending to the destination point. It's extensible allowing the HMI BL layer to be built on top of it and gives users a chance to select a BLs usage.

7.10.5 Deployment model

Figure 193 describes the logical deployment of the flexible bus lane application onto the CVIS infrastructure.

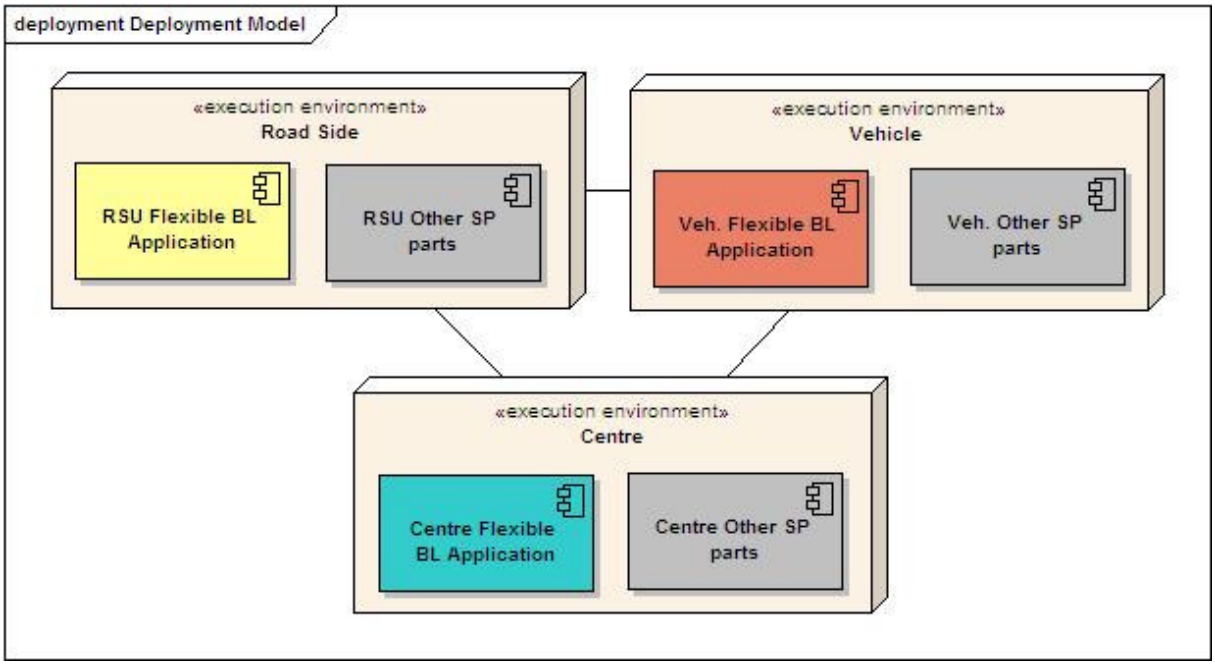

Figure 193: UML deployment diagram

7.11 Network assessment

The "Network Assessment" application and its main services are introduced in this sub-section. Further details of the network assessment application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.11.1 Overview

The availability of the information from the vehicle is chance to improve traffic related basic services and possibly to have an improved middle-long term planning of the infrastructure itself.

The network assessment application is a tool that is aimed at measuring the performance of the network. In particular the network assessment is aimed to:

analyzing the current state of the system,

assess the quality of the state of the network,

perform an off-line historical analysis,

provide ability to compare performance with or without the control system working.

The availability of data from the vehicle is a new source of information, nowadays only the road infrastructure can provide traffic information. These new data (xFCD) from the vehicle represents:

new data at all,

extended coverage of already available data,

incremented reliability, since it will be a separate source for the same data.

Thus incorporating this data into traffic system will allow having more precise knowledge of the traffic status, for middle-long term planning, maintenance and tuning.

The application is aimed at identify anomaly of the network caused by, for example:

road congestion event,

inefficient intersection regulation/control,

traffic model issue.

The network assessment function is designed to be running on the central system, possibly in the same location of the traffic control system or the traffic manager system.

Main use cases and system boundary

The main use cases and the system boundary of the traffic control assessment application is depicted in Figure 194.

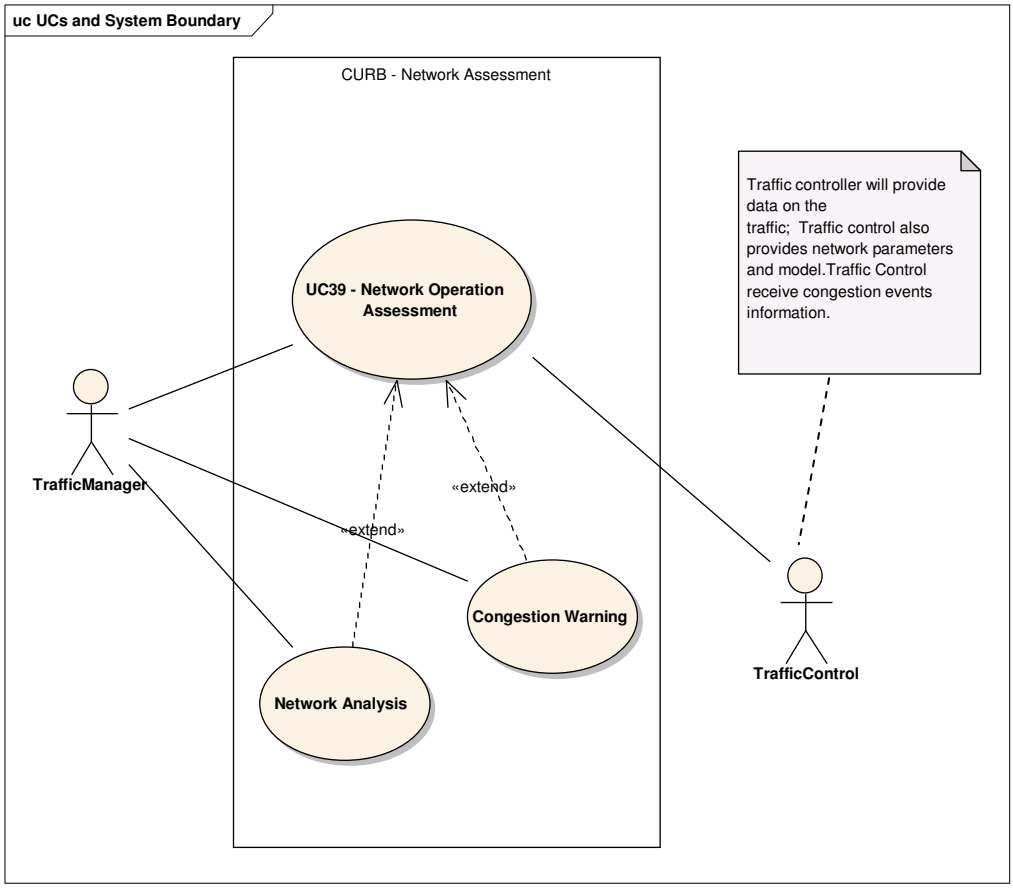

Figure 194: Use case model with system boundary

The use cases are as follows:

Network operation assessment: This use case describes the feature of assessing the working state of the network. This is a important tool for the traffic manager to decide on strategic infrastructure work, to tune the control system, to identify critical area of the network. Network assessment is aimed at real time and on-line monitoring of the network status, tuning of network parameters, network model assessment. The information is then used for congestion warning.

Congestion warning and **network analysis** are specialization of the network assessment UCs.

The actors and their needs and responsibilities are described in the following:

Traffic manager: Organisation responsible for maintaining the roads and managing the traffic on it. The traffic manager operator wants improve traffic control and traffic management by defining and implementing cooperative control and management strategies.

7.11.2 Application programming interface

The API of the traffic control assessment is depicted in Figure 195.

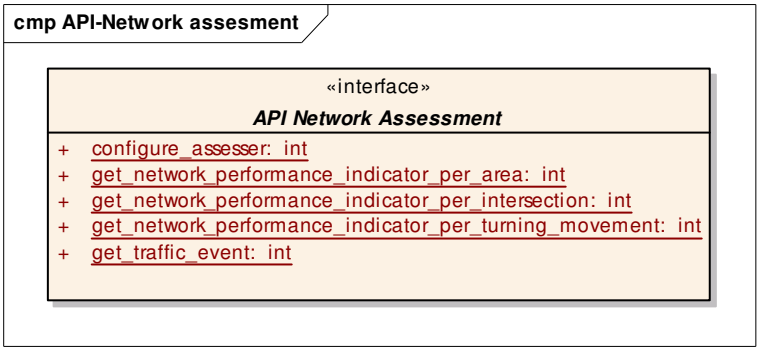

Figure 195: API of network assessment

The behavioural aspect of the API is illustrated with the example interactions below.

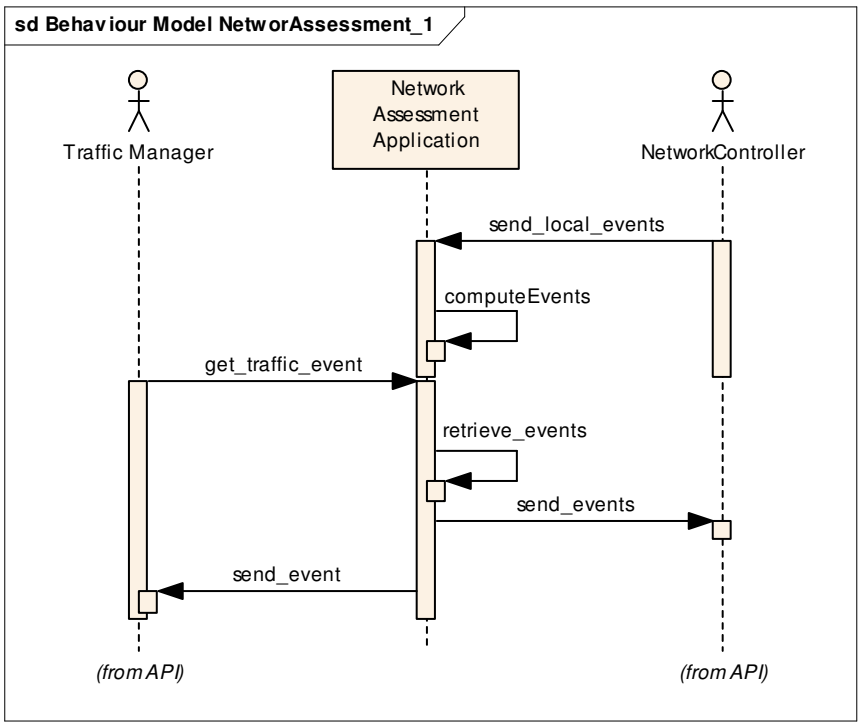

Figure 196: Behavioural model network assessment application

The network assessor will also provide information/indicator at the different level of the network and allow the configuration of the application as shown in the interaction model in Figure 197 .

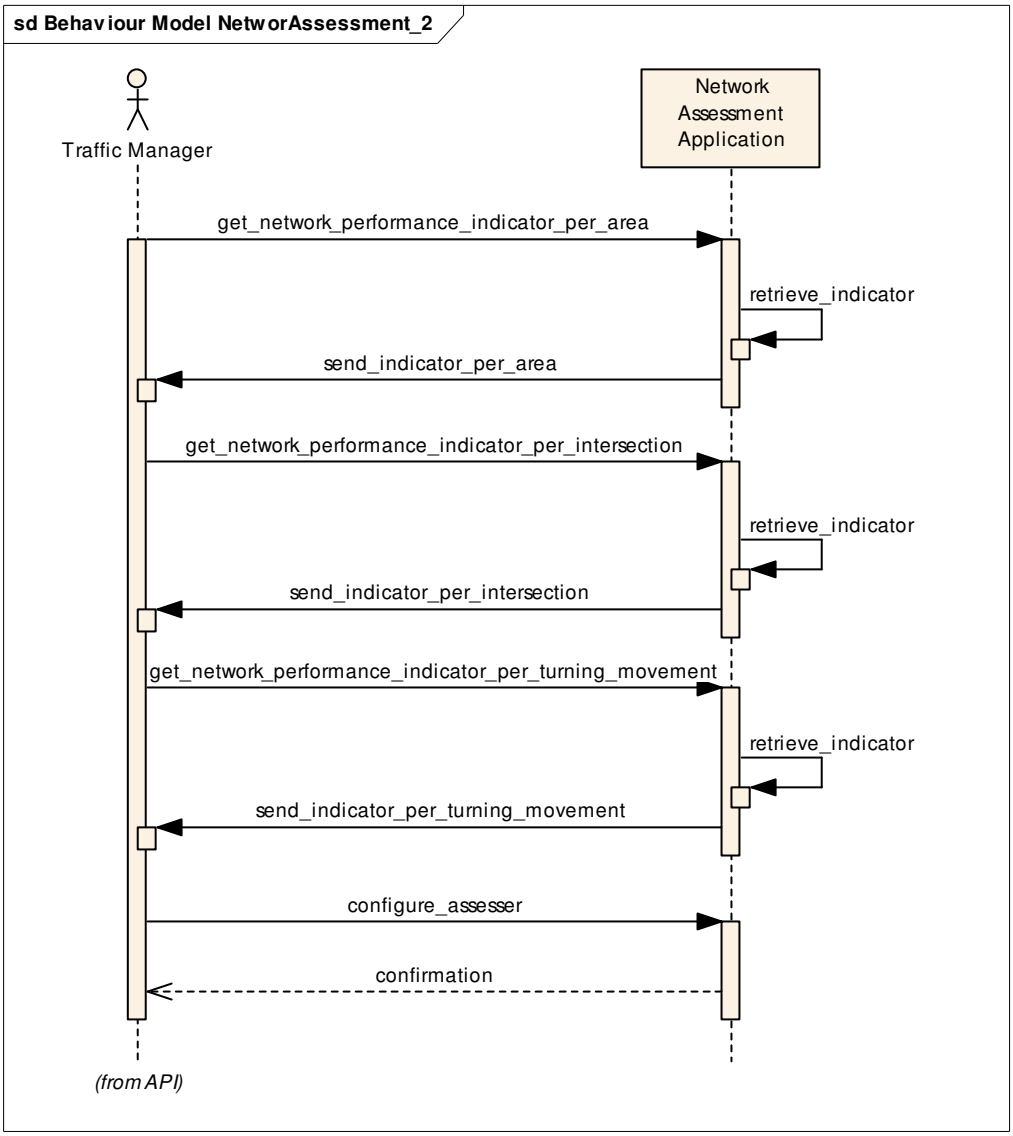

Figure 197: Behavioural model network assessment application

7.11.3 Information model

The information model of the "Traffic Control Assessment" application is depicted in Figure 198.

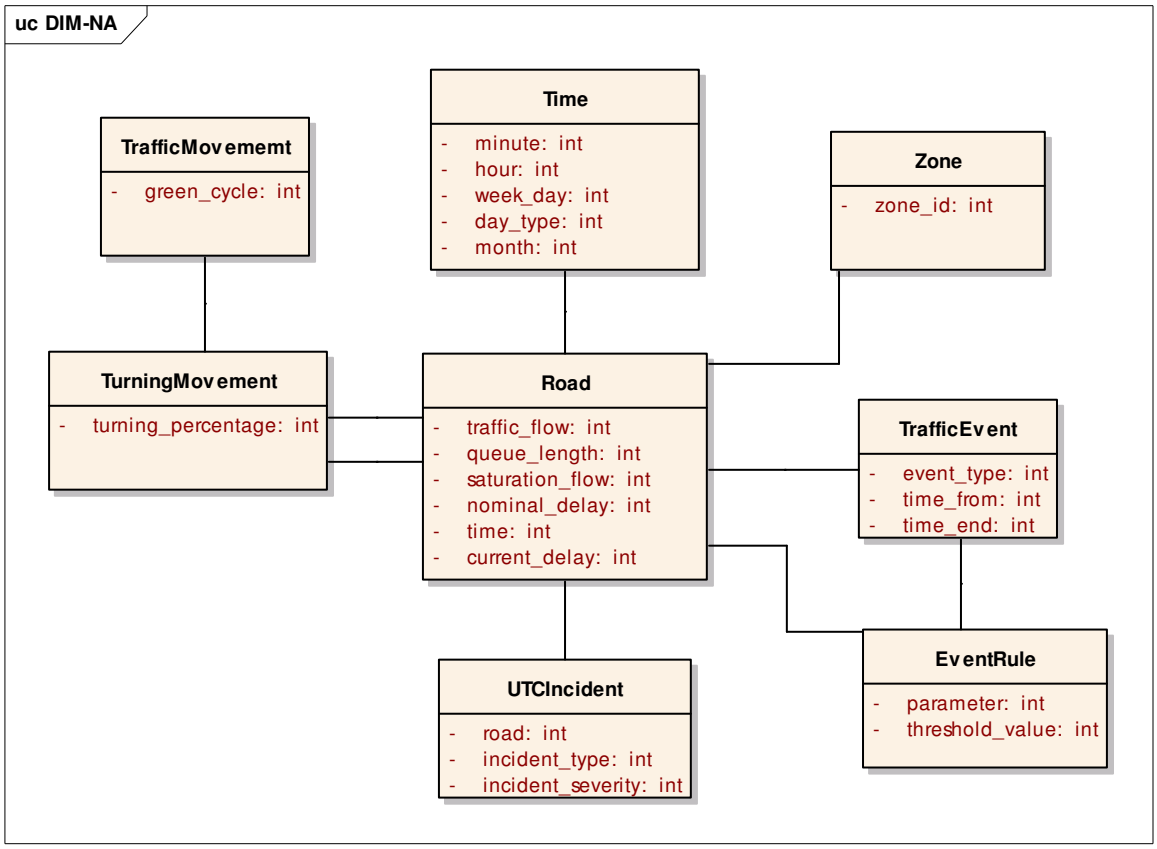

Figure 198: Domain information model network assessment application

In the Domain information model the following information is represented:

The static road network representation.

The traffic value as coming from the vehicle (xFCD) and RSU referred to the road.

Nominal status of the network.

Traffic events, as generated by the traffic assessment application.

Time dimension to represent historical data.

Event rule used to generate the traffic alarms.

Delay/flow information for historical trend analysis.

Incident coming from the UTC.

7.11.4 Interaction model

The overall interaction process of the network assessment application is specified in Figure 199.

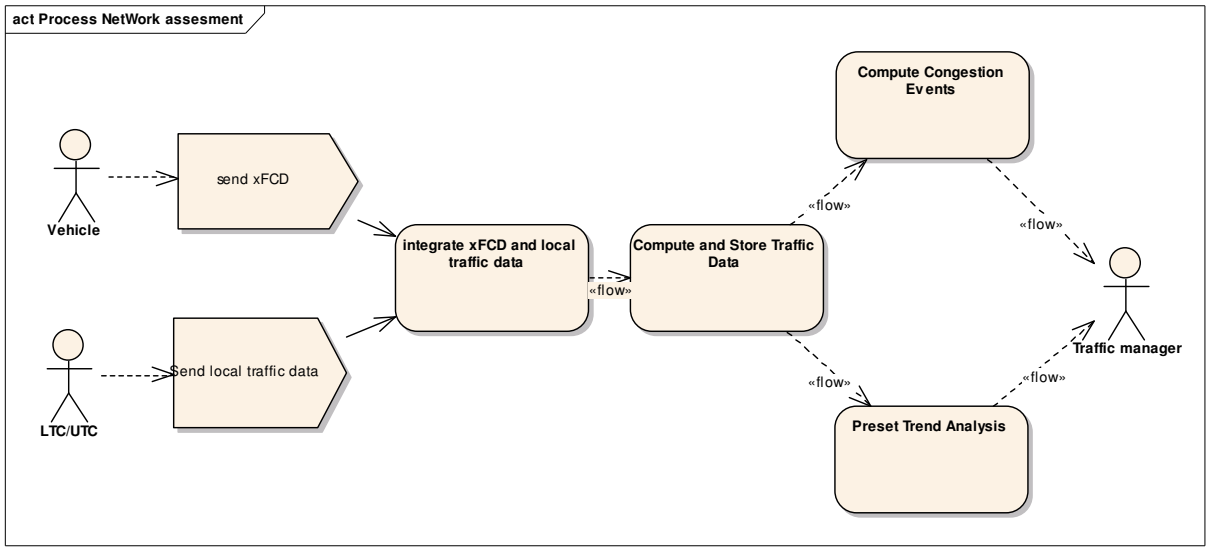

Figure 199: Network assessment application - overall process

Scenario description

This section describes the scenario for the network assessment application. The network assessment is a central application, thus it connects to the central. The cooperative traffic information facility data access manager (NDM as described in the cooperative traffic information facility document on architecture and specification). The network assessment (NA) client has access to the granted data for the whole urban network. The NA application also receives local incident information from the UTC/LTC. The data are integrated and processed in order to extract basic data, as delay-flow pairs. Data are then stored and presentation of delay-flow diagram or generation of traffic alarm can be performed.

To retrieve information from the RS, the NA shall either contact the central controller of the traffic control function or use the LDM/Message manager in place as defined in the cooperative traffic information facility.

In the following diagram the basic activity flow diagram is represented. The cooperative traffic information facility and UTC interaction are represented.

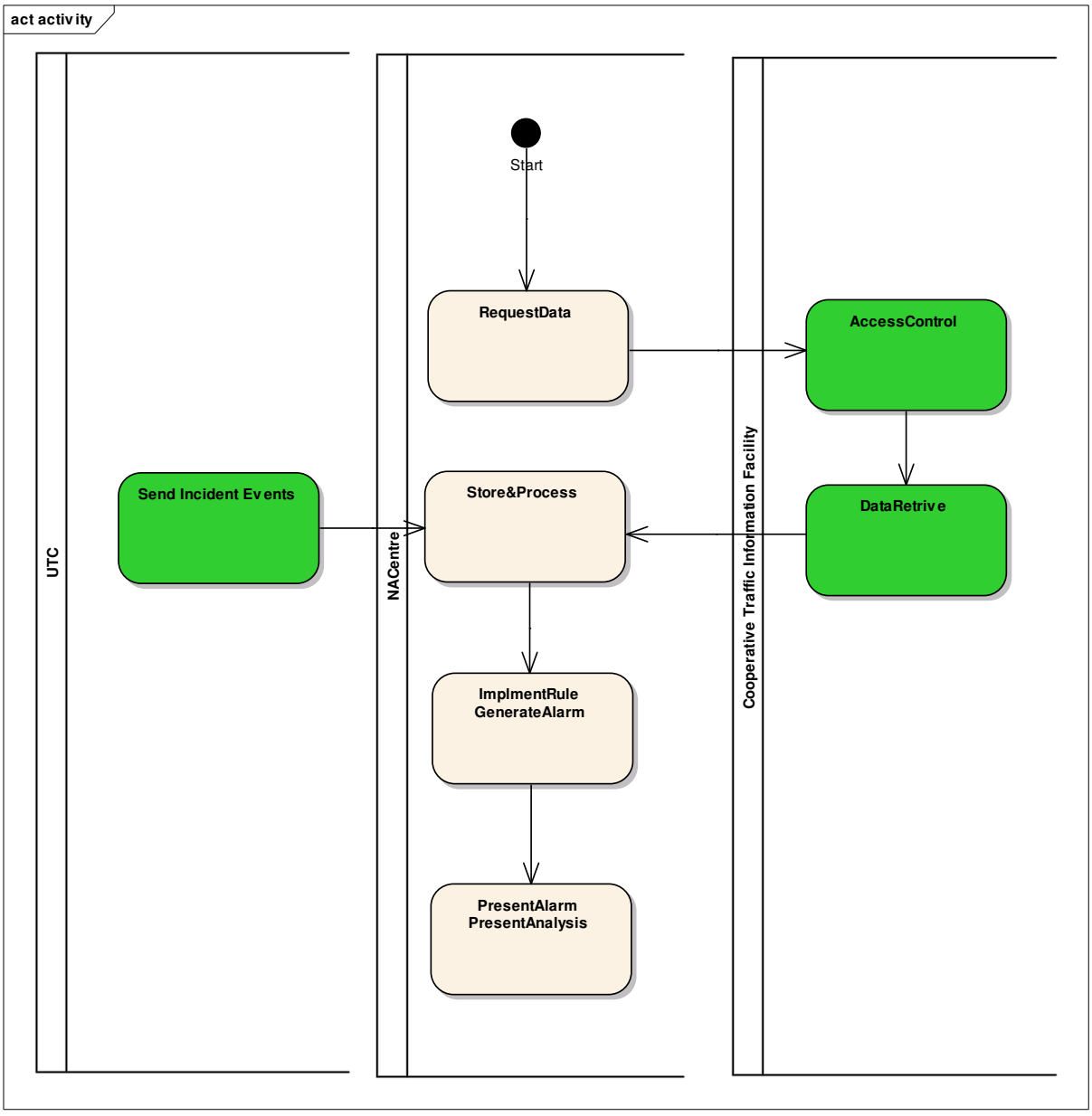

Figure 200: Network assessment activity diagram

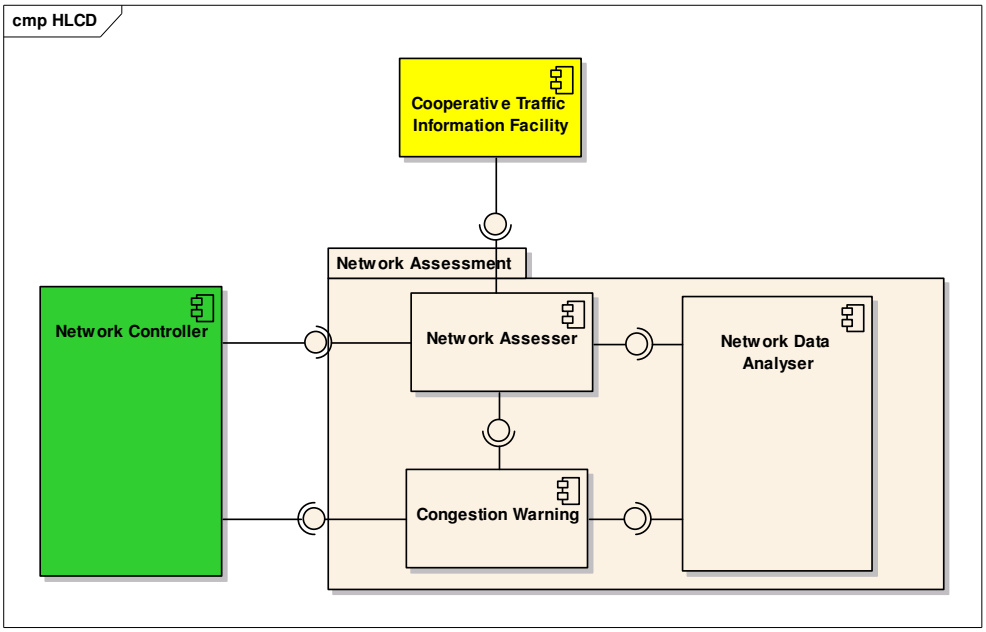

Figure 201: Component diagram of strategy application

Network assessor: The network assessor component is responsible to define the current state of the network and to store this information into the historical DB. This information is then used by analysis tool to allow prediction events, compare current and historical state.

Congestion warning: This component is responsible, based on the assessed state of the network, to detect anomaly. The component, assessing also the severity, can identify the cause and to define counter actions. The identification of the anomaly is based on rules.

Network data analyser: This component is the user interface to access the traffic assessment data. It provides information at link, area or network level, comparing historical information.

Cooperative traffic information NMD: It represents the access point to traffic data, as defined in the COMO Architecture and specification.

Network controller: It is the function of traffic control for the urban network.

7.11.5 Deployment model

The deployment diagram of the network assessment is depicted in Figure 202.

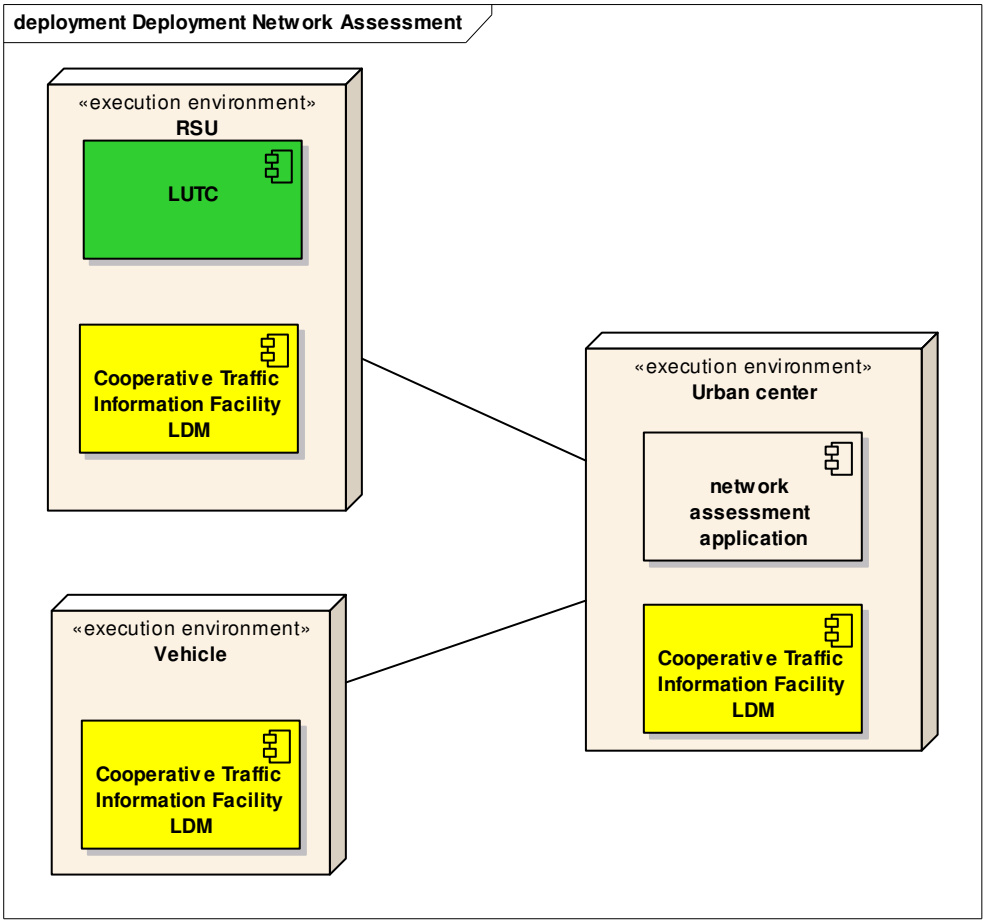

Figure 202: Network assessment deployment

7.12 Routing application

The routing application and its main services are introduced in this sub-section. Further details of the routing application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.12.1 Overview

The dynamic routing of drivers in the urban context aims at the reduction of congestion and travel time within the urban network, thus enabling a better use of the urban road network. While achieving this task, routing in the urban network shall also take into consideration other factors that are defined by the traffic manager or, more in general, by the local public authority. In particular the traffic manager sets the scenarios to be considered, which can depend on events in the city, e.g. football match, weather forecast or pollution consideration. The urban routing system receives then the strategy defined by the traffic management centre.

To perform the task of diverge drivers in the road network, the system shall transform the strategy provided using an appropriate way of geo-referencing into a route that the driver is

willing to follow. The only way to persuade the driver to follow the route is either because it is more convenient for him/her (in terms of travel time, cost, pollution, safety) or because some external enforcement. For the first case the driver shall be informed of the convenience of the route suggested. This information can be the expected travel time for the suggested route.

The application is aiming at providing suggestions of routes that take into consideration the strategy of the network and the willingness of following the route.

Another aspect of following the strategy of the network, and not provide a selfish solution, is that the other sub-systems of the traffic management system all act based on this strategy, so that the maximum of the objective can be achieved. The integration with the other urban sub-systems, notably collective routing and traffic control, would enable an integral strategy implementation.

Main use cases and system boundary

The main use cases and the system boundary are specified in Figure 203.

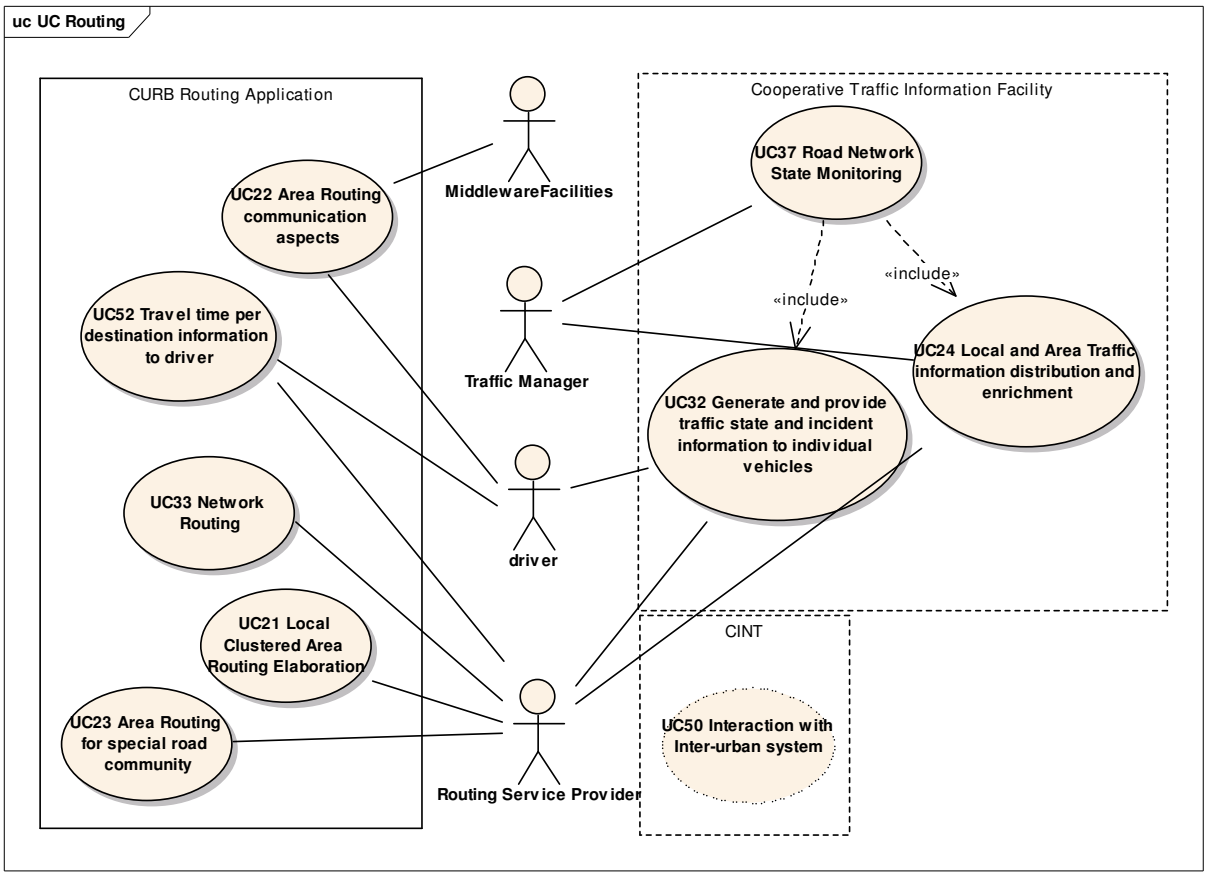

Figure 203: Use case model with system boundary

The use cases are as follows:

UC21 Local clustered area routing elaboration: The goal of this use case is to provide route guidance advice generation at local and distributed level. The area of influence can span from a single RSU, to a cluster of RSUs. The cluster can depend on the network and the actual position of the constraint. The information can then be propagated at higher level, to generate network level route guidance. The system will consider an incident or

momentary disturbance in traffic flow. By conducting the determination of incidents and the calculation of alternative routes within the local road-side infrastructure a highly responsive local area rerouting is realised. The private vehicle drivers benefit from a reduction in travel time and an increase of safety, while preserving network efficiency.

UC22 Area routing communication aspects: The goal of this use case is to provide : (i) the communication of the route options to the driver and (ii) the computation of route options based at infrastructure side based on the current locally available traffic information.

UC23 Area routing for special road community: The goal of the UC is the modelling and routing of special category driver and vehicle and the possibility to include special rules in the network description, as the presence of special road class or traffic limited zones.

UC24 Local and area traffic information distribution and enrichment: The goal of this use case is the distribution of network and local traffic situation and short and medium time prediction at network, area and local level. The information is supposed to come from higher level and to be eventually enriched with more detailed information, both from vehicle (private/commercial/public) and from conventional infrastructure based sensors.

UC32 Generate and provide traffic state and incident information to individual vehicles: The goal of this use case is to provide consistent high quality traffic information as on onboard service to the road user. The road user can request information about incidents and traffic state in certain urban networks or parts of urban networks. The information service is not connected with a routing service is therefore addressing certain user groups.

UC33 Network routing: The goal of this use case is to support the implementation of network wide traffic scenarios by giving routing advice to individual vehicles while knowing their characteristics and destination and also the network wide traffic conditions. By using individual information the road operator is able to offer, via a service provider, several individual route options that cannot be offered in a collective way, e.g. by road-side displays. The road operator can achieve better distribution of traffic by routing vehicles via links with capacity resources.

UC37 Road network state monitoring: This use case is aimed at: i) Feeding traffic planning assessment, for demand management and environmental assessment, ii) Feeding network traffic operation assessment, for incident management, iii) Feeding traffic control strategy assessment, for designing, tuning or operation of traffic control strategy

UC52 Travel time per destination information to driver Actors involved are the driver of the CVIS equipped vehicle, traffic management centres that provide the traffic information and the road-side units in the CVIS network that distribute local traffic information.

The actors and their needs and responsibilities are described in the following:

Driver: The private driver wants to travel through the urban network in a comfortable, safe and efficient way. He is interested in being supported by dynamic information and navigation systems.

Middleware facilities: Represents the CVIS basic and domain facilities (see part II of this document)

Traffic manager: The traffic manager is responsible for managing the traffic in the urban network. The traffic manager aims to reach a safe and fluent traffic flow in the urban network by informing the road users and controlling the traffic flow. He tries to achieve an increase of safety in the urban network by implementing due to the current traffic situation certain cooperative traffic management scenarios. The traffic manager uses CVIS services in the traffic management centre.

Routing service provider: The service provider procures services including the corresponding content. The service provider represents a business related entity. He supplies the necessary means to provide the business related support of a specific service application. He is also responsible for delegating the task of service deployment.

7.12.2 Application programming interface

There are three interfaces provided by the routing application

Driver interface

Information interface

Data Exchange interface

These are specified in the following driver interface:

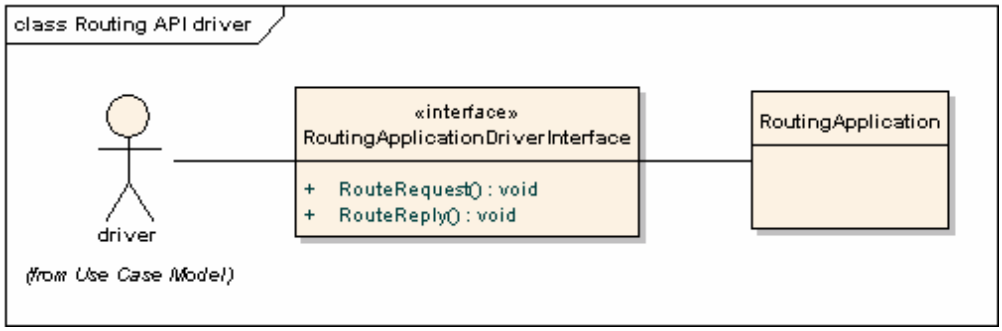

Figure 204: API driver interface

For the RouteRequest method the following information is provided:

destination,

start point,

Vehicle type,

Vehicle ID,

route preferences.

For the RouteReply method the following information is received:

Reference route (waypoints) and route related traffic information:

- incident type,
- position,
- travel speed,
- travel times.

The behavioural aspect of the API is illustrated with the example interaction in Figure 205.

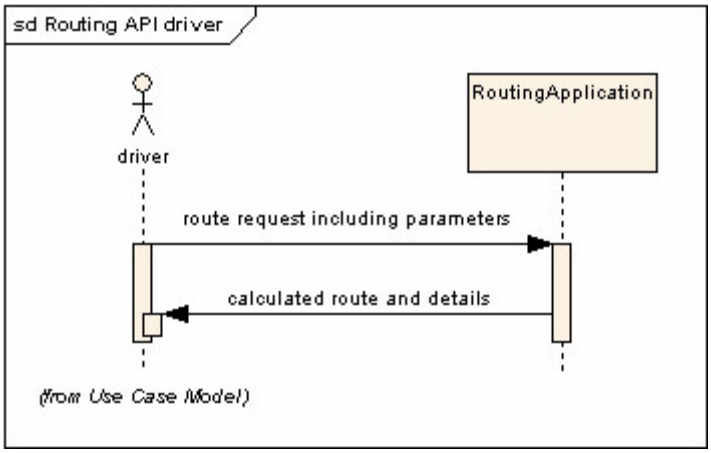

Figure 205: Behavioural model routing application driver interface

Information interface

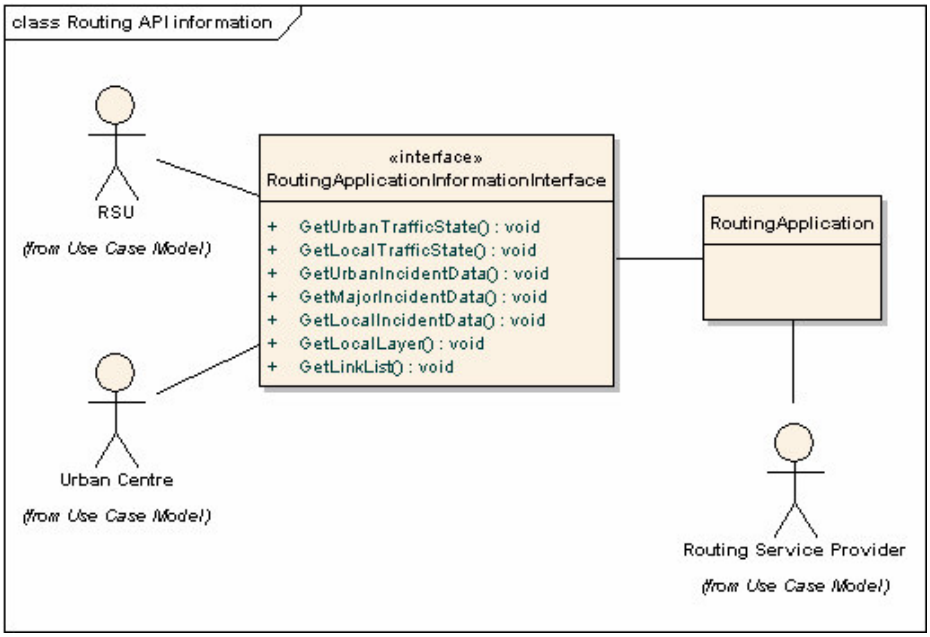

Figure 206: API information interface

The methods are elaborated further below:

For the *GetUrbanTrafficState* and *GetLocalTrafficState* the following information on urban or local traffic state is received:

travel time,

travel speed,

level of service in the urban network.

All information is geo-referenced. Reference route (waypoints) and route related traffic information:

For the *GetUrbanIncidentData, GetMajorIncidentData* and *GetLocalIncidentData* the

following information on urban or local incidents is received (major incidents implies information on incidents that might have a high input on traffic flow in the urban network):

incident type,

position,

starting time,

expected duration travel time.

For the *GetLocalLayert* the following information is received:

Information on local strategy to be considered by the vehicle routing engine. The local strategy can either be defined using waypoints or by transferring the affected area sections and their impedance factors.

For the *GetLinkList* the following information is received:

links,

allocated impedance factors.

High impedance factors are allocated to the links of the congested main routes and low impedance factors are allocated to the links of the prioritized alternative routes.

The behavioural aspect of the API is illustrated with the example interaction in Figure 207.

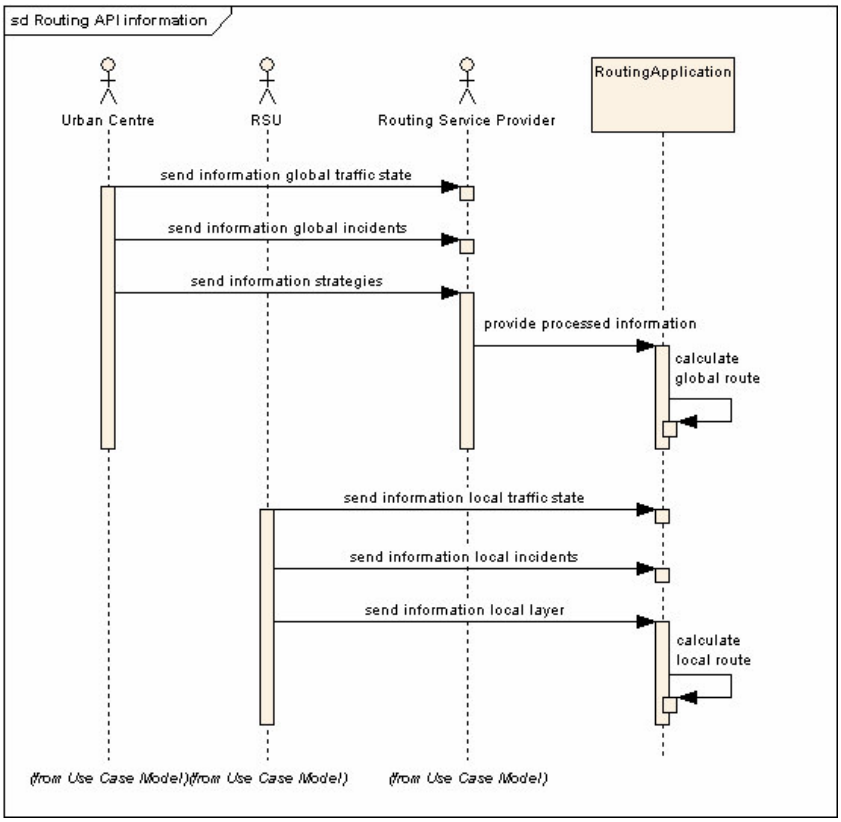

Figure 207: Behavioural model routing application information interface

Data exchange interface

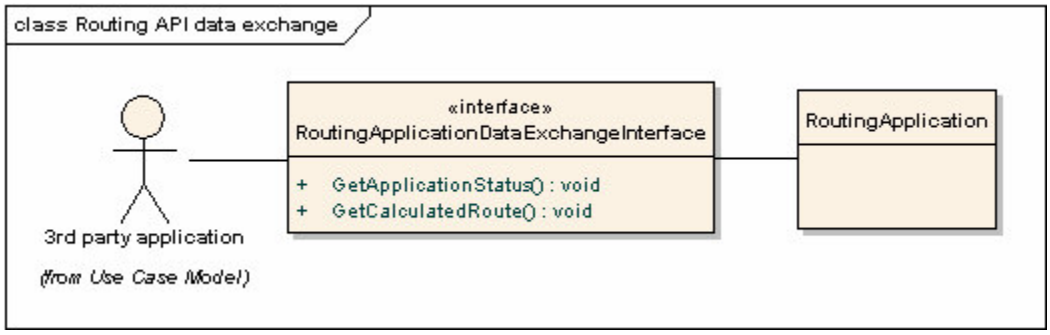

Figure 208: API data exchange interface

The behavioural aspect of the API is illustrated with the example interaction in Figure 209.

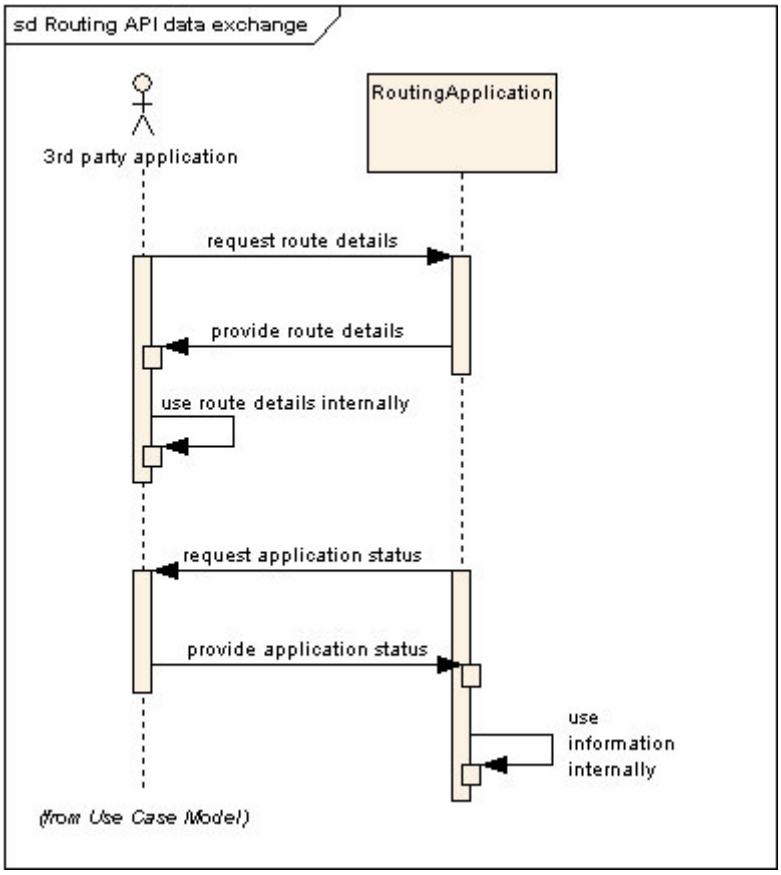

Figure 209*:* Behavioural model routing application data exchange interface

7.12.3 Information model

This section provides the specification of the information model. The information model identifies and defines the main concepts of the application domain. The concepts are specified in terms of their types using UML class diagram as shown in Figure 210.

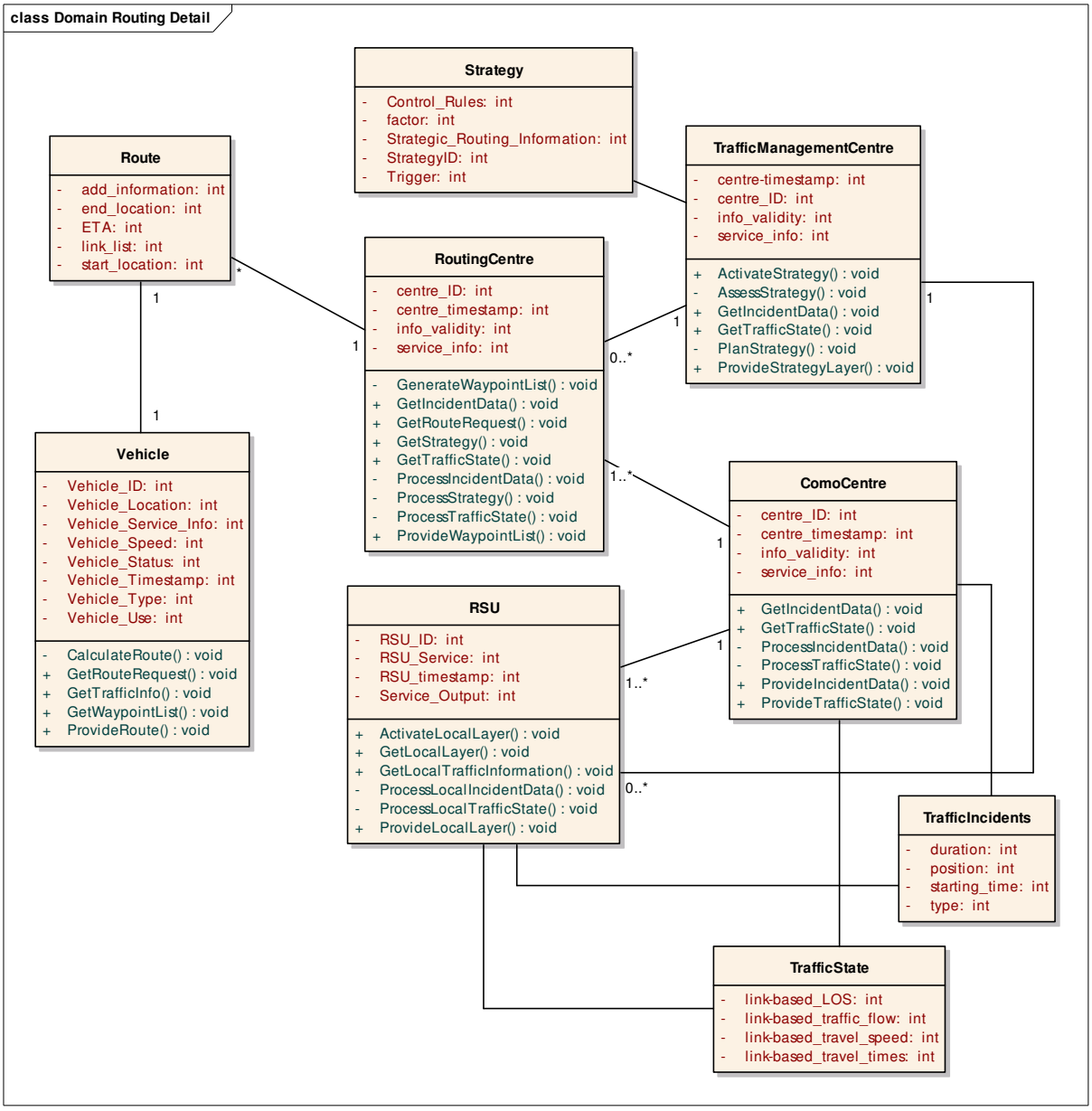

Figure 210: Domain information model routing application

7.12.4 Interaction model

The sequence of events of the routing application is as follows:

1. The driver starts the vehicle; automatically the CVIS system connects to the CVIS network.
2. The driver starts his navigation system and enters his destination and route criteria via HMI into the system.
3. Based upon the driver's input and also the current vehicle position and vehicle specific information, e.g. vehicle type, the in-vehicle CVIS system generates a route request.
4. This route request is transmitted to the CVIS service centre for further processing.
5. Furthermore the service centre also receives current traffic and incident data as well as activated public strategies from the urban centre.
6. Based upon this information as well as on the driver's route request the service centre calculates a reference route (way point list) and transmits it to the in-vehicle routing system for further processing. In addition also route related traffic information, e.g. congestion, accident, road works, are transmitted to the in-vehicle routing system.
7. The in-vehicle routing system analyses the received reference route and calculates the precise route, i.e. the best route for the driver's request.
8. The calculated route as well as route related traffic information is communicated to the driver via HMI interface.
9. The road-side units continuously inform the surrounding area about their presence and the available services.
10. While driving, the CVIS vehicle connects to a nearby RSU that provides the needed routing functionality.
11. If applicable, the in-vehicle CVIS system receives from the RSU an activated local strategy (activation occurs based upon local traffic and incident data). Furthermore the in-vehicle CVIS system also receives relevant traffic and incident information form the RSU.
12. Based upon the activated local strategy the in-vehicle CVIS system recalculates the route; the route update is communicated to the driver via HMI.
13. If there is no RSU available/nearby for a certain time interval while driving, the in-vehicle CVIS system automatically connects to the CVIS service centre and asks for a route update.

The overall interaction process is depicted in Figure 211.

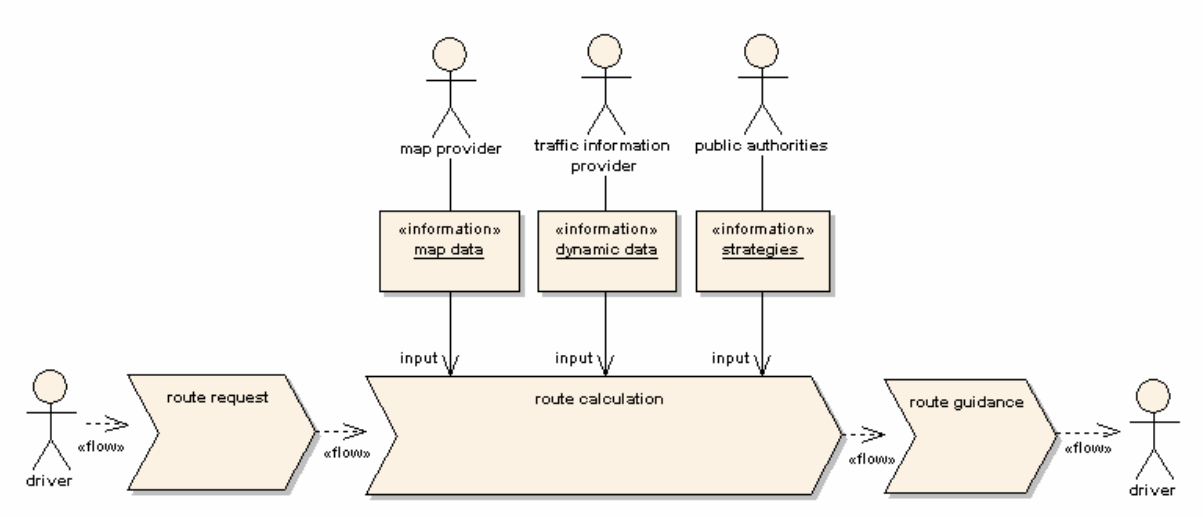

Figure 211: Reference service process routing application

7.12.5 High level composite architecture

This section provides specification of the high level composite architecture. The high level composite architecture describes the overall architecture of the system and the partitioning into sub-systems and components. It also identifies the interfaces and the relationships between the sub-systems, components and interfaces.

The high level composite architecture of the routing application is shown in Figure 212.

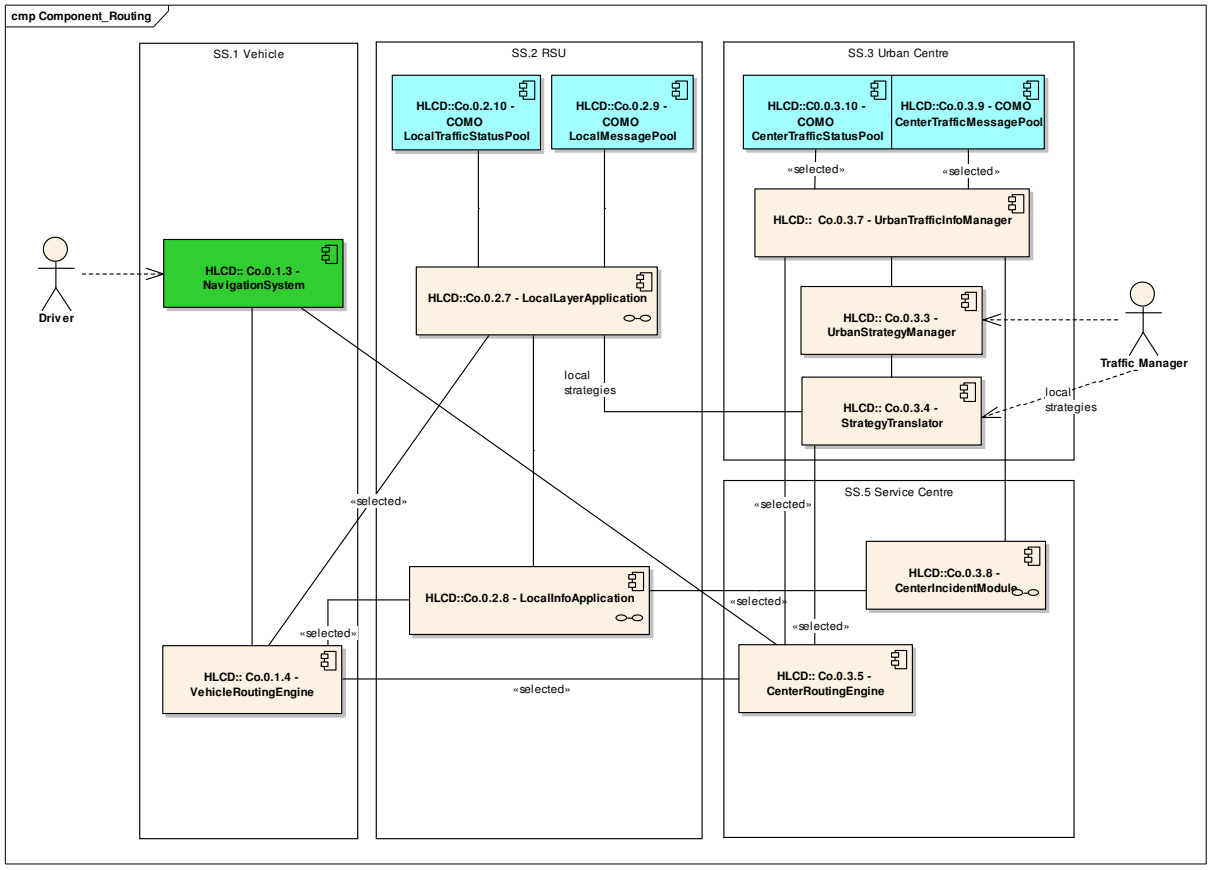

Figure 212: High level composite architecture routing application

HLCD:: Co.0.2.7 Local layer application This high level component is composed of three sub-components whose functionalities are described in the following:

Local traffic info manager: This component receives local traffic info (traffic state & incident data) from the cooperative traffic information facility. This component transmits local traffic info to the local decision maker (to be considered for the activation of local layers). This component transmits local traffic info to the local info application (to be distributed to the vehicle).

Local layer DB: This component stores and provides predefined local layers. Local layers are allocated to certain local traffic situations and consist of impedance factors for local routes. Local congested main routes shall be avoided and local alternative routes shall be prioritised by the vehicle routing engine.

Local decision maker: The local decision maker decides, based on the local traffic info received from local traffic info manager, to activate a certain local layer (stored in the local layer database). The activated local layer is transmitted to the vehicle routing engine for further route calculation.

HLCD:: Co.0.2.8 LocalInfoApplication This high level component is composed of two sub-components whose functionalities are described in the following:

LocalIncidentModule: This component receives major incidents, e.g. congestion, road works, road blocking, from the service centre incident module.

LocalInfoProvider: This component receives local traffic info from the local traffic manager (sub component of the local layer application) as well as major incidents from the incident module and transmits both kind of information to the vehicle routing engine for further processing.

HLCD:: Co.0.3.8 CenterIncidentModule

This high level component is composed of two sub-components whose functionalities are described in the following:

IncidentEvaluator This component recognizes major incidents (based on the information received from the urban traffic info manager) and defines an area of interest where this information should be distributed (via local RSUs).

IncidentDistributor:This component distributes the information on major incidents to the local incident modules (sub-component of the local info application)

HLCD:: Co.0.3.9 COMO centre traffic message pool: This cooperative traffic information facility component provides incident data event triggered to the urban traffic info manager.

HLCD:: Co.0.3.20 COMO centre traffic status pool: This cooperative traffic information facility component provides a traffic state report for the urban network in fixed time intervals to the urban traffic info manager.

HLCD:: Co.0.2.9 COMO local message pool: This cooperative traffic information facility component provides local incident data event triggered to the local traffic info manager.

HLCD:: Co.0.2.10 COMO local traffic status pool: This cooperative traffic information facility component provides a traffic state report for the local urban network in fixed time intervals to the local traffic info manager.

HLCD:: Co.0.1.3.NavigationSystem: The navigation system translates the precise route (received by the vehicle routing engine) to a human readable information. The navigation system generates the route request (based on driver's input) or the navigation system interacts with the driver to define the destination.

Optionally the navigation system will interact directly with the centre routing to request/display the route and to interact to proceed in the route. The off board navigation is a client application that guides the user to his/her selected destination. Routes and maps are downloaded dynamically from the centre router only when needed, covering the current user position surroundings; therefore there is no need to have all the cartography stored on the device. All information is updated to the current traffic situation and to updated cartography. Maps are in SVG tiny format, a W3C standard for 2D vector graphics, especially suited for mobile application with limited processing and memory capabilities. The user can have, besides a visual navigation, turn by turn indications that are downloaded together with the route.

The off board navigation will get the GNSS position either by itself or from an external device application with which can be integrated as an external library.

HLCD:: Co.0.1.4.VehicleRoutingEngine The component transmits the route request (received form the navigation system) to the centre routing engine. The component receives the reference route (way point list) and route related traffic info from the service centre (centre routing engine). The reference route (way point list) and traffic information are processed within the vehicle routing engine. Result is the precise route and additional

information, e.g. travel times, incident information, to be communicated to the driver (via navigation system).The component receives the local layer (if activated) and local traffic info for local route update. Furthermore also major incidents are received.

HLCD:: Co.0.3.3.UrbanStrategyManager This component stores, handles and activates public strategies that are then transmitted to the centre routing engine (via strategy translator) to be considered for the route calculation. The adequate strategies are selected by evaluating information on the traffic status of the urban network (received from the urban traffic info manager).

HLCD:: Co.0.3.4.StrategyTranslator This component receives public strategies from the urban strategy manager and translates them into routing information, e.g. link list, to be considered by the centre routing engine. There are three options for this component: i) the strategy is transmitted as a list of link, ii)the strategy is transferred as user equilibrium solution, or iii) the strategy translator has an active role and reply for the only urban area to request of the central router with waypoints for the requested route.

HLCD:: Co.0.3.5.CentreRoutingEngine

The component receives route requests from vehicle routing engine. Therefore it calculates the "reference route" (way point list) based on the received traffic info (current traffic state & incident data) and strategies (if activated). The reference route and route related traffic info are sent back to the on-board routing component (vehicle routing engine) for further processing.

HLCD:: Co.0.3.7.UrbanTrafficInfoManager

This component receives current urban network traffic state (fixed time intervals) and incident data (event triggered). The information is transmitted to:

Urban strategy manager (for selection of the adequate strategy),

Centre routing engine (for dynamic route calculation),

Incident evaluator (for recognition of major incidents to be distributed to the local RSUs).

7.12.6 Deployment model

See Figure 212.

7.13 Strategy application

The strategy application and its main services are introduced in this sub-section. Further details of the strategy application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.13.1 Overview

On top of traffic control, the traffic demand management system has the task to define the strategy to implement in the traffic network. There could be various approaches to define the traffic assignment. The two main solutions to this problem currently present are:

1. design of the strategy based on traffic status and prediction, traffic demand pattern and statistical assignment,

2. centralised selection of pre-defined strategy based on traffic status or traffic status simulation or prediction.

These two schemes are based on the assumption that there is a scenario of operation that is either selected by the traffic management operator, e.g. for special events, or in an automatic fashion, e.g. base on period of the year or calendar.

The traffic demand management system gets information on historical data of traffic, on traffic operator preferences and provides commands to the traffic control system and the routing system. The system can also interact with public transport and parking systems.

Main use cases and system boundary

The main use cases and the system boundary is specified in Figure 213.

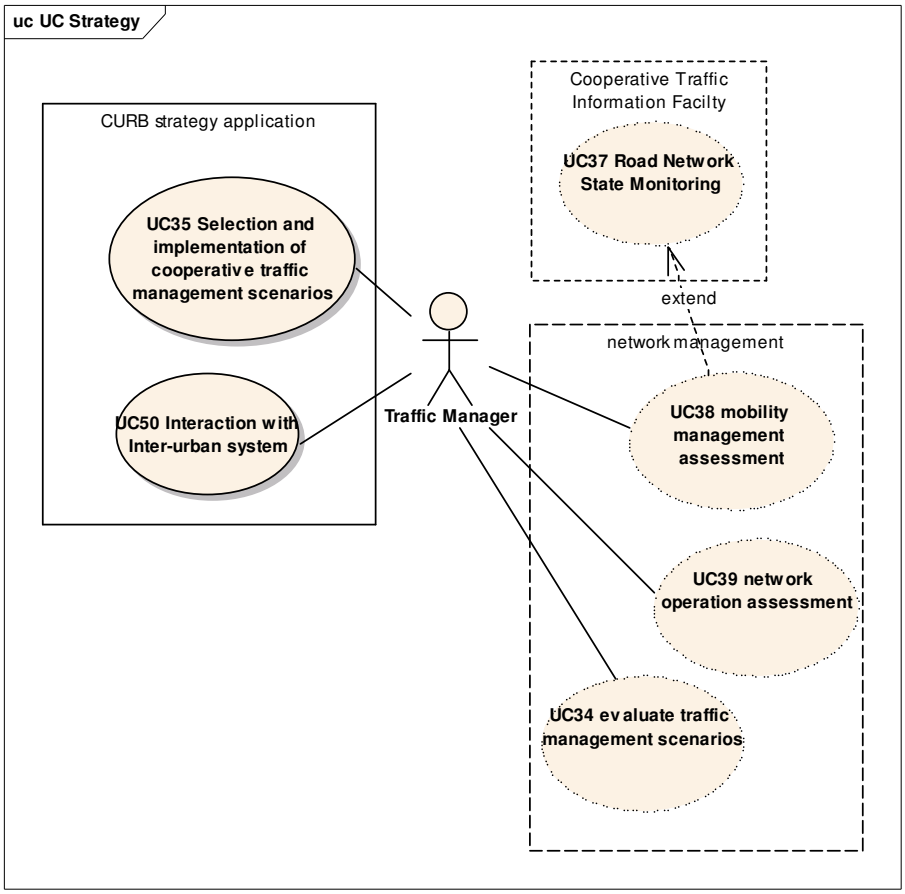

Figure 213: Use case model with system boundary

The use cases are as follows:

UC34 Evaluate traffic management scenarios: The goal of this use case is to evaluate traffic management scenarios and to offer the road operator and the traffic manager to prove, if the expected effects have been reached by the implementation of a certain scenario. The operator is able to analyse the quality of the implemented scenarios in an off-line process based on historical data. Based on the evaluation road authorities can find resources in the network, plan traffic management scenarios and prove the reasonability of the given routing advices. The operator is able to modify measures in the simulation model and to evaluate the effect of the modifications. A result of this process may be a list of

measures to be modified within the defined cooperative traffic management scenarios. The evaluation of traffic management scenarios in an off-line simulation does not cover all the main aspects of a cooperative scenario, but it is a prerequisite to provide a highly effective cooperative traffic management system.

UC35 Selection and implementation of cooperative traffic management scenarios: The goal of this use case is to select and implement network wide cooperative urban traffic management scenarios based on the current network situation. The selection of the optimal traffic management scenario is performed by a CVIS control model and an on-line simulation model. The control model proposes a scenario from a list of pre-defined scenarios. The scenarios consist of: i) control rules to be implemented in the urban network, ii) information, i.e. on incidents, to be distributed via different media, iii) and routes or corridors prioritised by the road operator to be provided to certain road user groups A simulation model predicts the effects of the proposed scenario and the current scenario and based on the results a decision is taken, if the current scenario retains unchanged or the proposed scenario is going to be implemented.

UC37 Road network state monitoring: This use case is aimed at: i) feeding traffic planning assessment, for demand management and environmental assessment, ii)feeding network traffic operation assessment, for incident management, and iii) feeding traffic control strategy assessment, for designing, tuning or operation of traffic control strategy.

UC38 Mobility management assessment: The goal of the UC is to obtain the possibility to tune the transport models and to assess the traffic models. This ability is used by the traffic manager and local authority for mobility and demand management. Different type of information are usually integrated e.g. population, origin-destination pattern, public transport planning data. Information, from vehicle and from conventional sources shall be integrated and simulated in batch mode, i.e. without the need to be on-line.

UC39 Network operation assessment: The use case describes the real time and on-line monitoring of the network status, the tuning of network parameters and the network model assessment. The extracted information can be used for incident detection, congestion warning possibly accident detection. The information is important for transport manager, traffic operator and road users. The first to assess the operation of the network and possibly to manage incident, the later to possibly avoid congested areas.

The actors and their needs and responsibilities are described in the following:

Traffic manager: The traffic manager is responsible for managing the traffic in the urban network. The traffic manager aims to reach a safe and fluent traffic flow in the urban network by informing the road users and controlling the traffic flow. He tries to achieve an increase of safety in the urban network by implementing due to the current traffic situation certain cooperative traffic management scenarios. The traffic manager uses CVIS services in the traffic management centre.

7.13.2 Application programming interface

The API of the flexible bus lane application is specified in Figure 214.

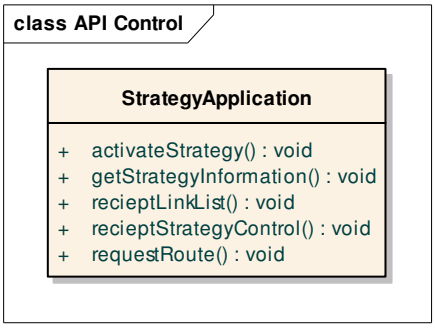

Figure 214: API Strategy application interface

The behavioural aspects of the provided methods are illustrated by means of the example interactions with the routing application and network controller in the figures below.

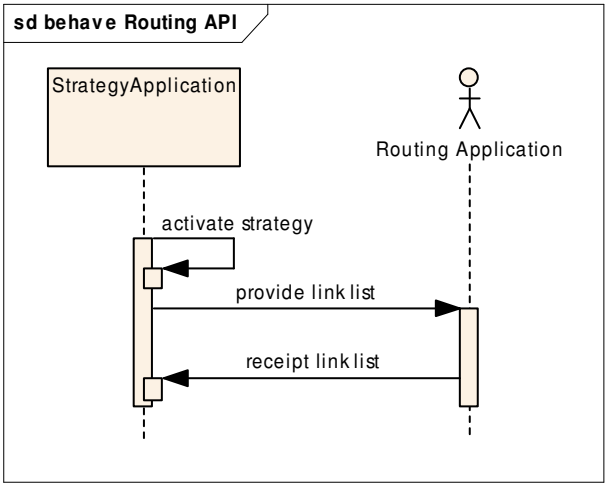

Figure 215: Behavioural model strategy application routing interface

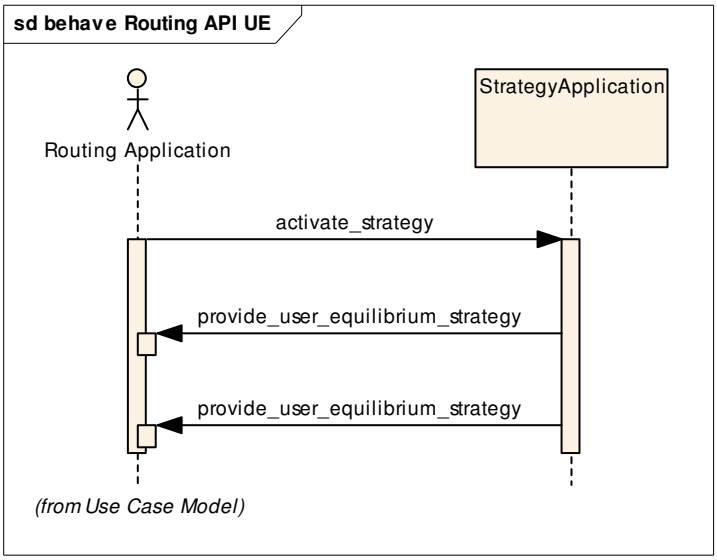

Figure 216: Behavioural model strategy application routing interface (user equilibrium)

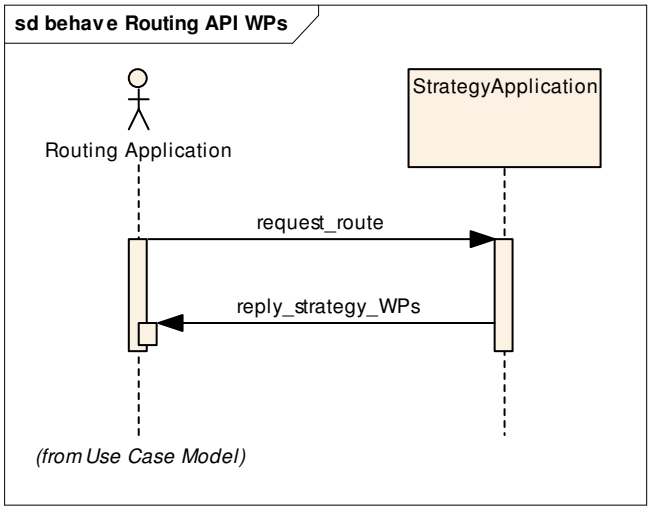

Figure 217: Behavioural model strategy application routing interface - WPs

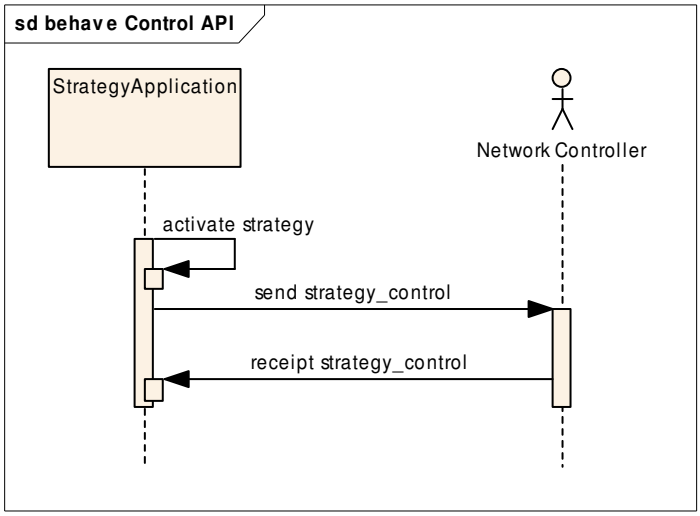

Figure 218: Behavioural model strategy controller interface

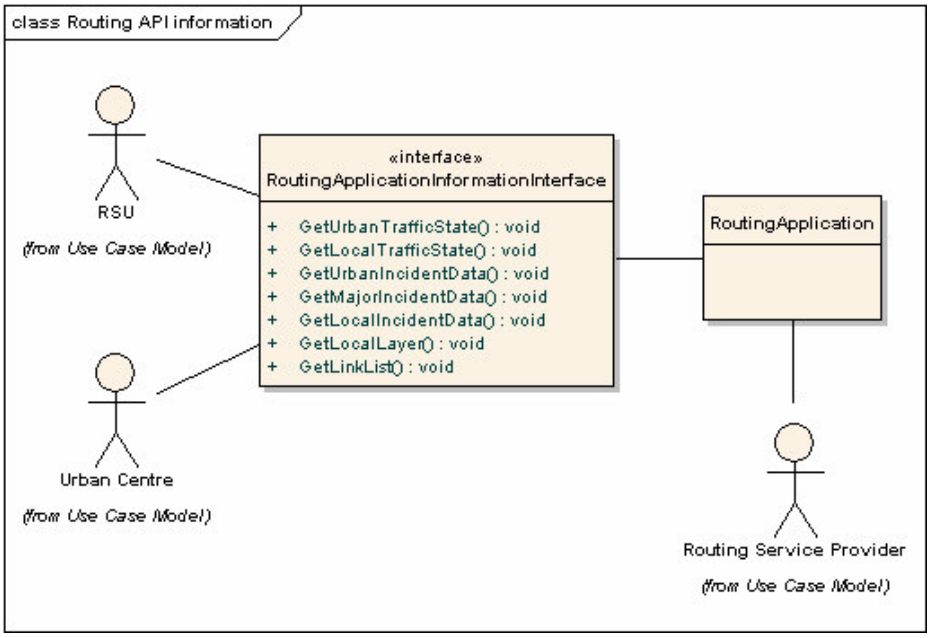

Figure 219: API information interface

7.13.3 Information model

This section provides the specification of the information model. The information model identifies and defines the main concepts of the application domain. The concepts are specified in terms of their types using UML class diagram as shown in Figure 220.

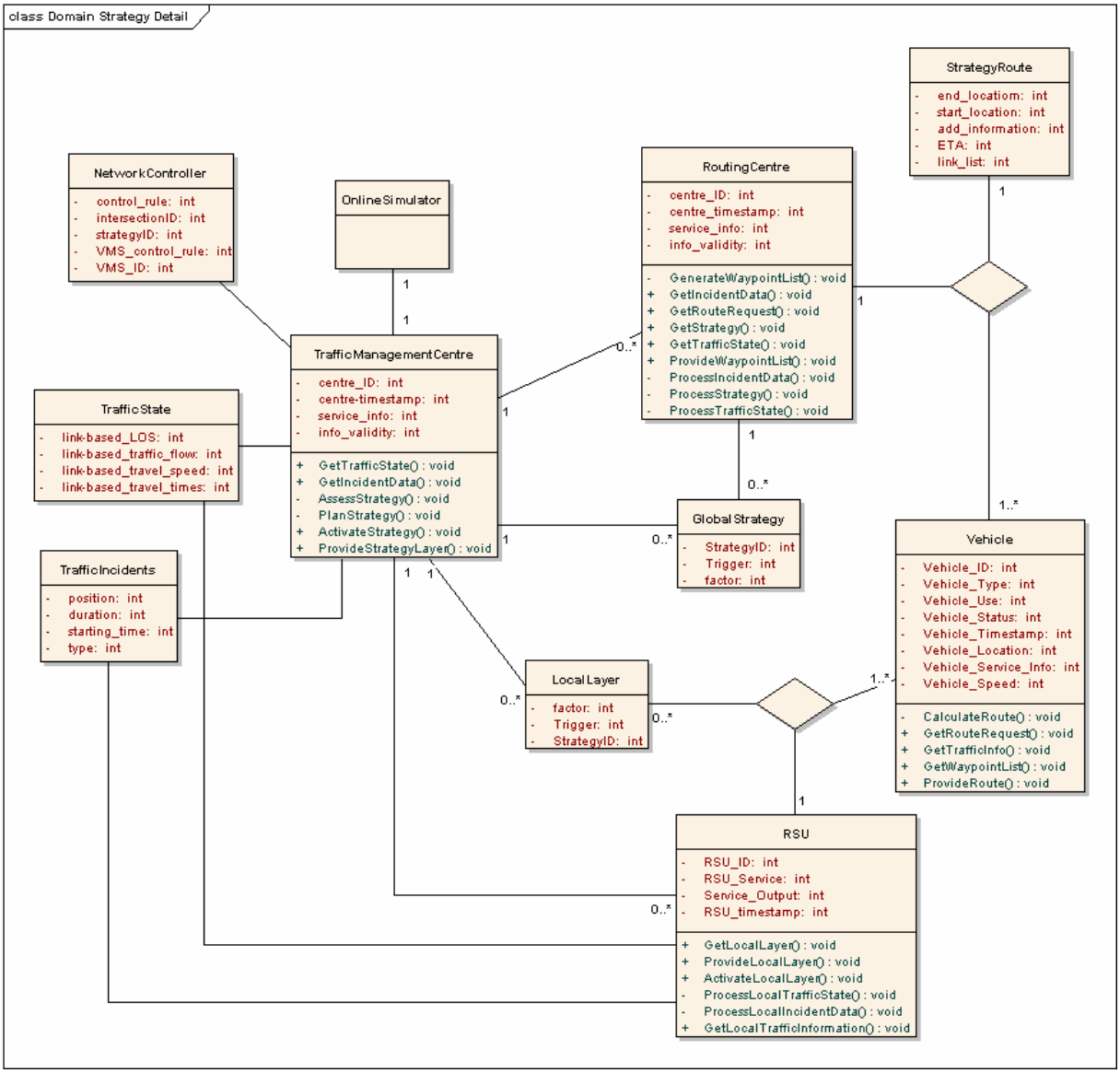

Figure 220: Information model of strategy application - link list

7.13.4 Interaction model

The sequence of events of the strategy application is as follows:

1. The urban traffic info manager situated in the urban centre receives urban network wide traffic state and urban incident data from COMO.
2. Incident data and traffic state are transferred to the strategy manager and optional to an online simulation model.
3. Based on the current traffic situation the strategy manager chooses an adequate strategy from a bunch of pre-defined strategies.
4. The control rules of the chosen strategy are implemented by a network controller.
5. The strategy information is provided to the service provider of routing services to be considered when calculating routes for the urban network.

Optional:

3.a The chosen strategy is simulated in online-simulation.

3.b The online-simulation predicts the traffic quality for the next time intervals assuming the implementation of the chosen strategy.

3.c Information about predicted traffic quality is provided to the strategy manager. The strategy manager decides if the traffic quality is adequate or if another strategy shall be implemented.

The overall interaction process is depicted in Figure 221.

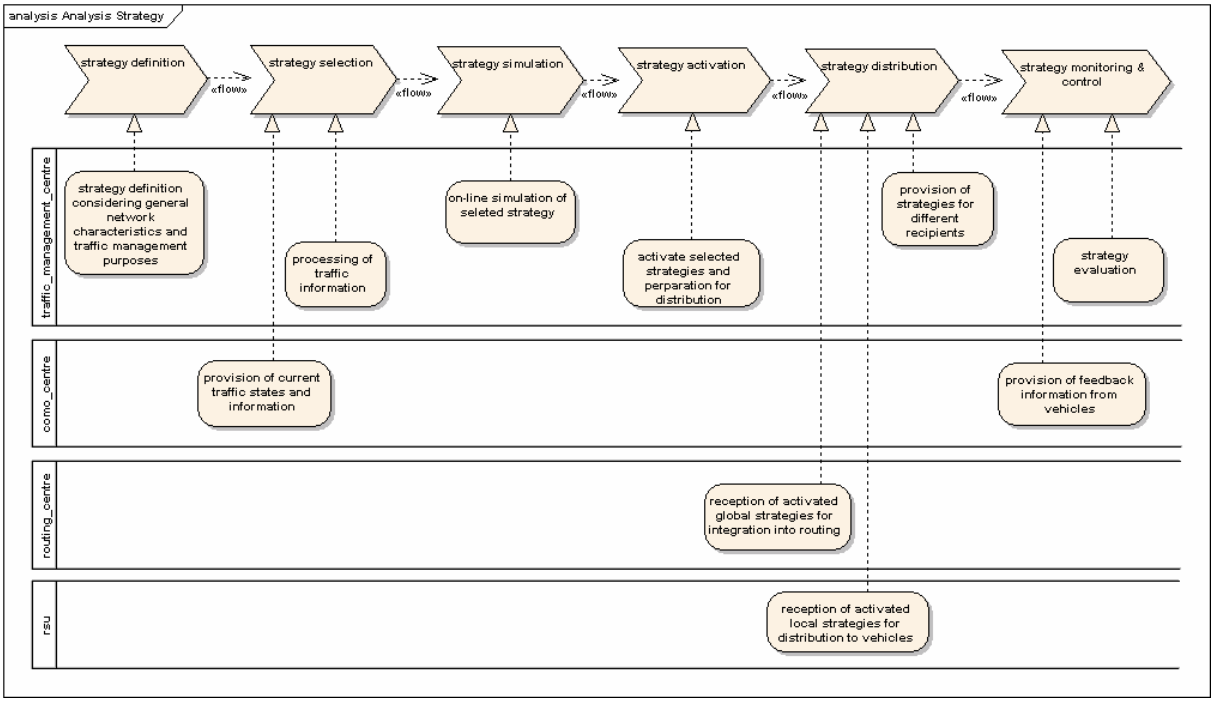

Figure 221: Reference service process strategy application

7.13.5 High level composite architecture

This section provides specification of the high level composite architecture. The high level composite architecture describes the overall architecture of the system and the partitioning into sub-systems and components. It also identifies the interfaces and the relationships between the sub-systems, components and interfaces.Figure 222.

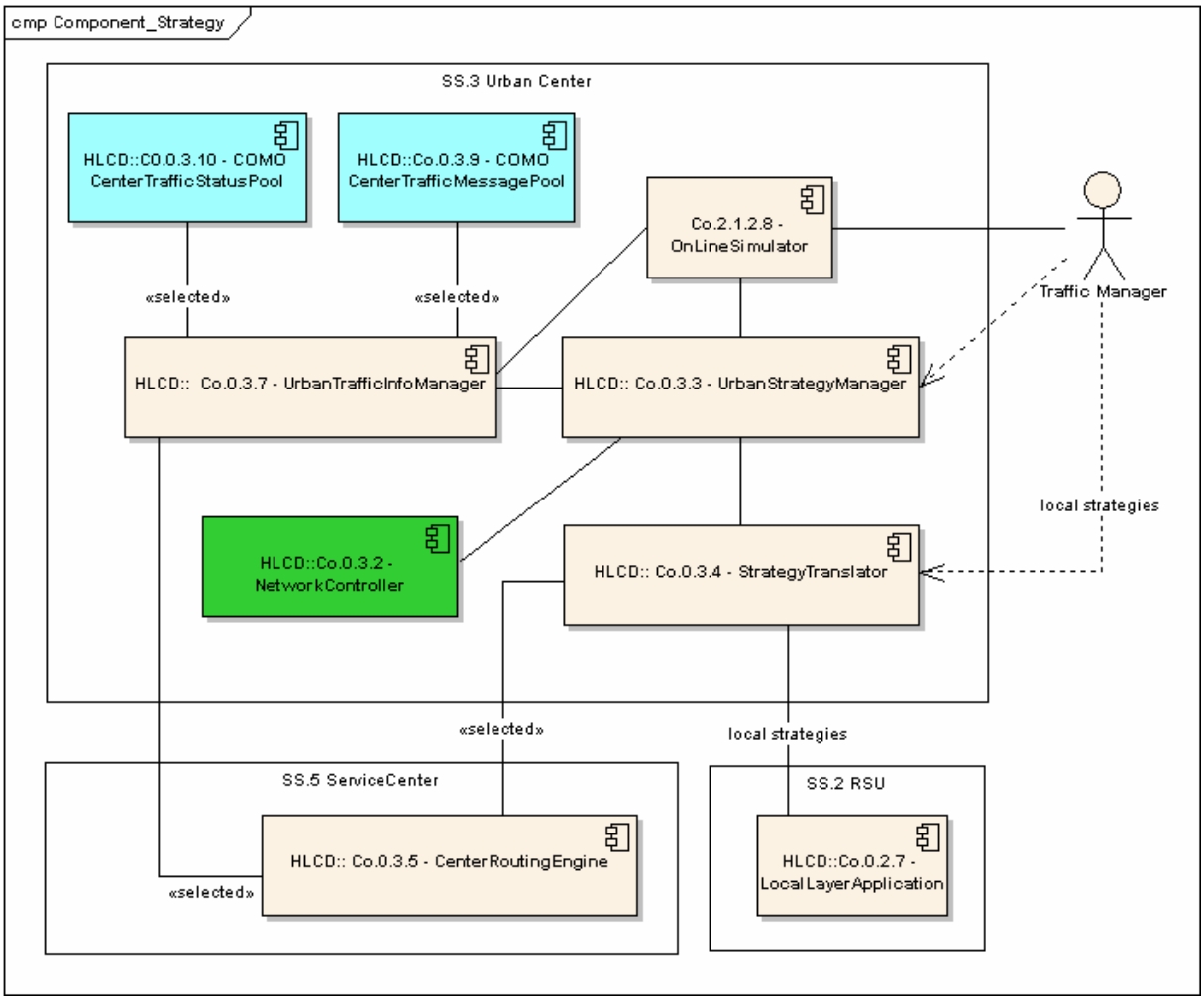

Figure 222: Component diagram of strategy application

The components are elaborated below:

Co.2.1.2.8. On-line simulator: This component receives current traffic status (incident data and traffic state) from the urban traffic info manager as well as selected strategies from the urban strategy manager. The on-line simulator predicts the forthcoming traffic status implying the selected strategies and the current traffic situation. The predicted traffic state is assessed and a quality factor for every selected strategy is returned to the urban strategy manager.

HLCD:: Co.0.2.7 Local layer application: This high level component is composed of three sub-components: i) local traffic, ii) info manager, iii) local layer DB

HLCD:: Co.0.3.2 Network controller: The network controller receives information on control rules (based on strategies) for traffic lights and VMS to be implemented.

HLCD:: Co.0.3.9 Traffic message pool: This cooperative traffic information facility

component provides incident data event triggered to the urban traffic info manager.

HLCD:: Co.0.3.20 Traffic status pool: This cooperative traffic information facility component provides a traffic state report for the urban network in fixed time intervals to the urban traffic info manager.

HLCD::Co.0.3.3. Urban strategy manager: This component stores, handles and activates public strategies that are then transmitted to the centre routing engine (via strategy translator) to be considered for the route calculation as well as to the network controller. The adequate strategies are selected by evaluating information on the traffic status of the urban network (received from the urban traffic info manager). The selected strategies are transmitted to the online simulator to be assessed regarding their estimated impacts on the traffic state and their usefulness. The on-line simulator assesses the selected strategies and returns a corresponding quality factor. Based on this quality factor the urban strategy manager decides on the activation of the strategy. If decision is positive, the strategy is provided to the centre routing engine.

HLCD:: Co.0.3.4 Strategy translator: This component receives activated public strategies from the urban strategy manager and translates them into routing information, e.g. link list, to be considered by the routing engine. There are three options for this component: i) the strategy is transmitted as a list of link, ii) the strategy is transferred as user equilibrium solution or iii) the strategy translator has an active role and reply for the only urban area to request of the central router with waypoints for the requested route.

HLCD:: Co.0.3.5 Centre routing engine: This component receives route requests from vehicle routing engine. Therefore it calculates the "reference route" (way point list) based on the received traffic info (current traffic state & incident data) and strategies (if activated). The reference route and route related traffic info are sent back to the on-board routing component (vehicle routing engine) for further processing.

HLCD:: Co.0.3.7 Urban traffic info manager: This component receives current urban network traffic state (fixed time intervals) and incident data (event triggered) from COMO. COMO information is transmitted to:

urban strategy manager (for selection of the adequate strategy),

online simulator (for strategy assessment).

7.13.6 Deployment model

The strategy application deployment diagram is specified in Figure 223.

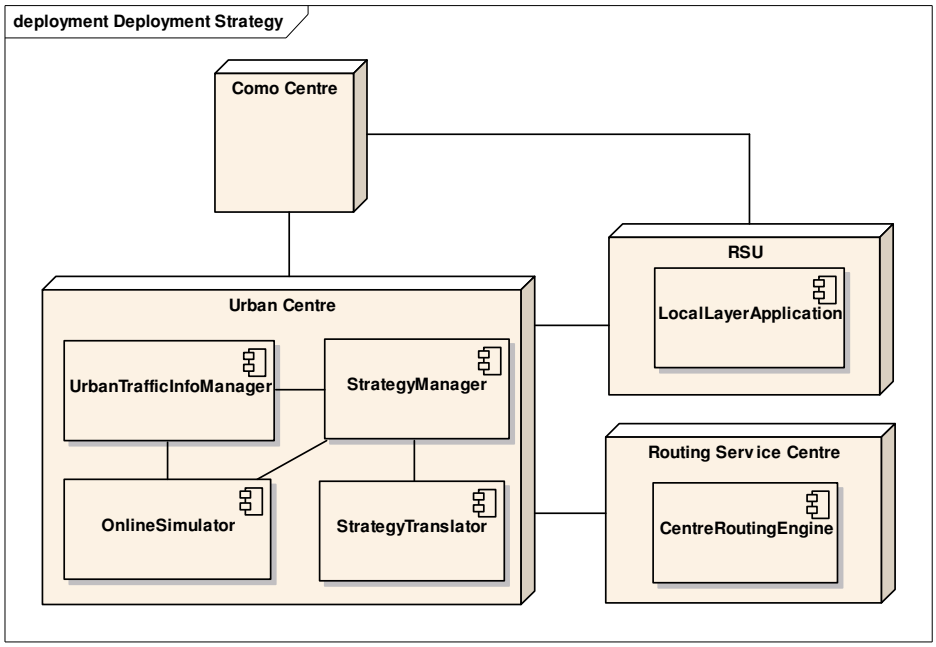

Figure 223: Strategy application deployment diagram

7.14 Traffic control assessment

The "Traffic Control Assessment" application and its main services are introduced in this sub-section. Further details of the traffic control assessment application can be found in the D.CURB.3.2 "Architecture Specification" document.

7.14.1 Overview

This application is aimed at assessing the traffic model and to estimate local traffic model parameters. It is meant to be a local application that runs in parallel to the traffic control sub-system and that allow the system to tune the parameters, estimate dynamic parameters and possibly to assess the behaviour of the local traffic controller.

The dynamic parameters to be estimated are, for example:

turning percentage,

clearance capacity.

These parameters are estimated based on the data coming from the vehicle and local available data. The local available data are:

traffic volumes,

density,

speed,

counting,

O/D.

All this data are estimated from both vehicle data and local present information.

Main use cases and system boundary

The main use cases and the system boundary of the traffic control assessment application is depicted in Figure 224.

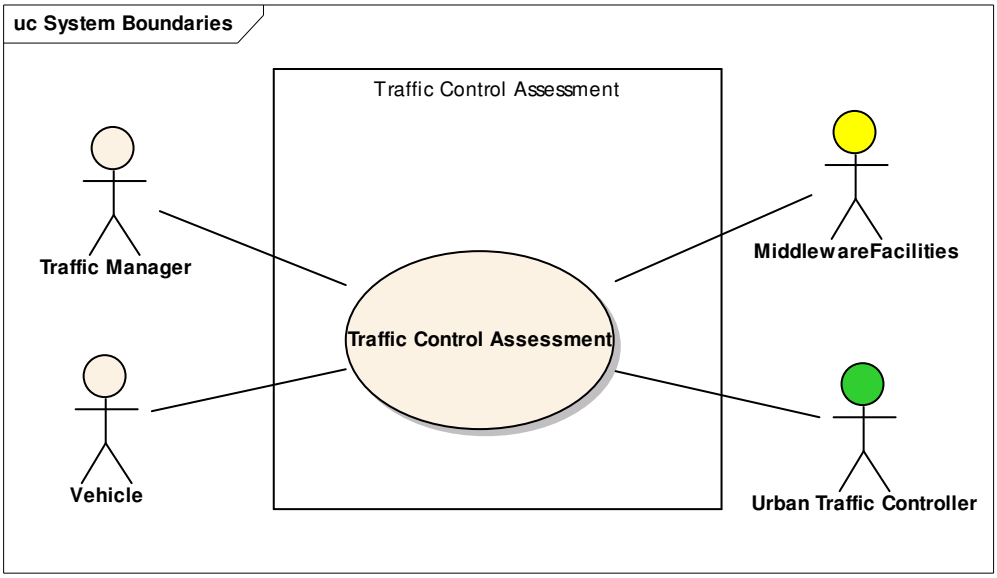

Figure 224: Use case model with system boundary

The use cases are as follows:

UC Traffic control assessment: The aim of the UC is to feed data into control model, assess and tuning the control model. Thus the monitoring data are used for traffic control tuning, design and operation.

The actors and their needs and responsibilities are described in the following:

Traffic manager represents the organization responsible for maintaining the roads and managing the traffic on it. The traffic manager wants improve traffic control and traffic management by defining and implementing cooperative control and management strategies.

Vehicle provides location information etc to be processed by the traffic control assessment use case.

Urban traffic controller maintain the data describing the current local traffic situation.

Middleware facilities represent the CVIS basic and domain facilities (see part II of this document).

7.14.2 Application programming interface

The API of the traffic control assessment application is depicted in Figure 225.

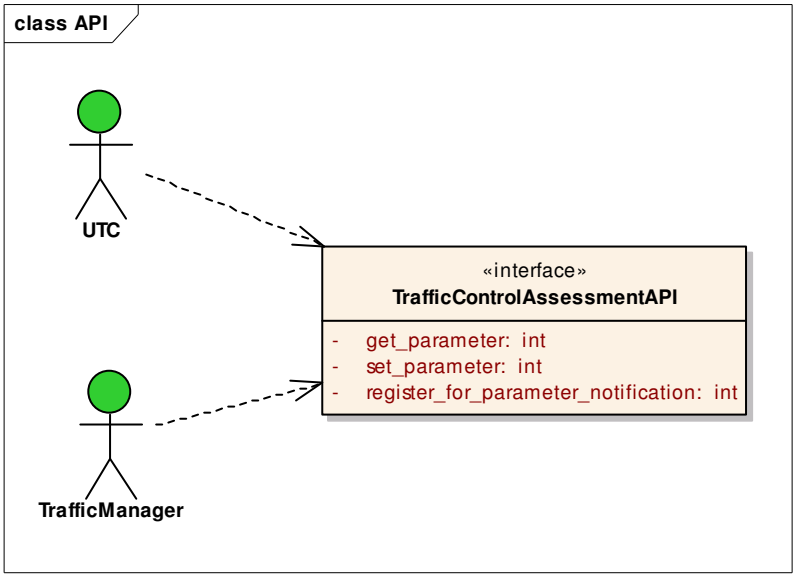

Figure 225: API of traffic control assessment

7.14.3 Information model

The information model of the traffic control assessment application is depicted in Figure 226.

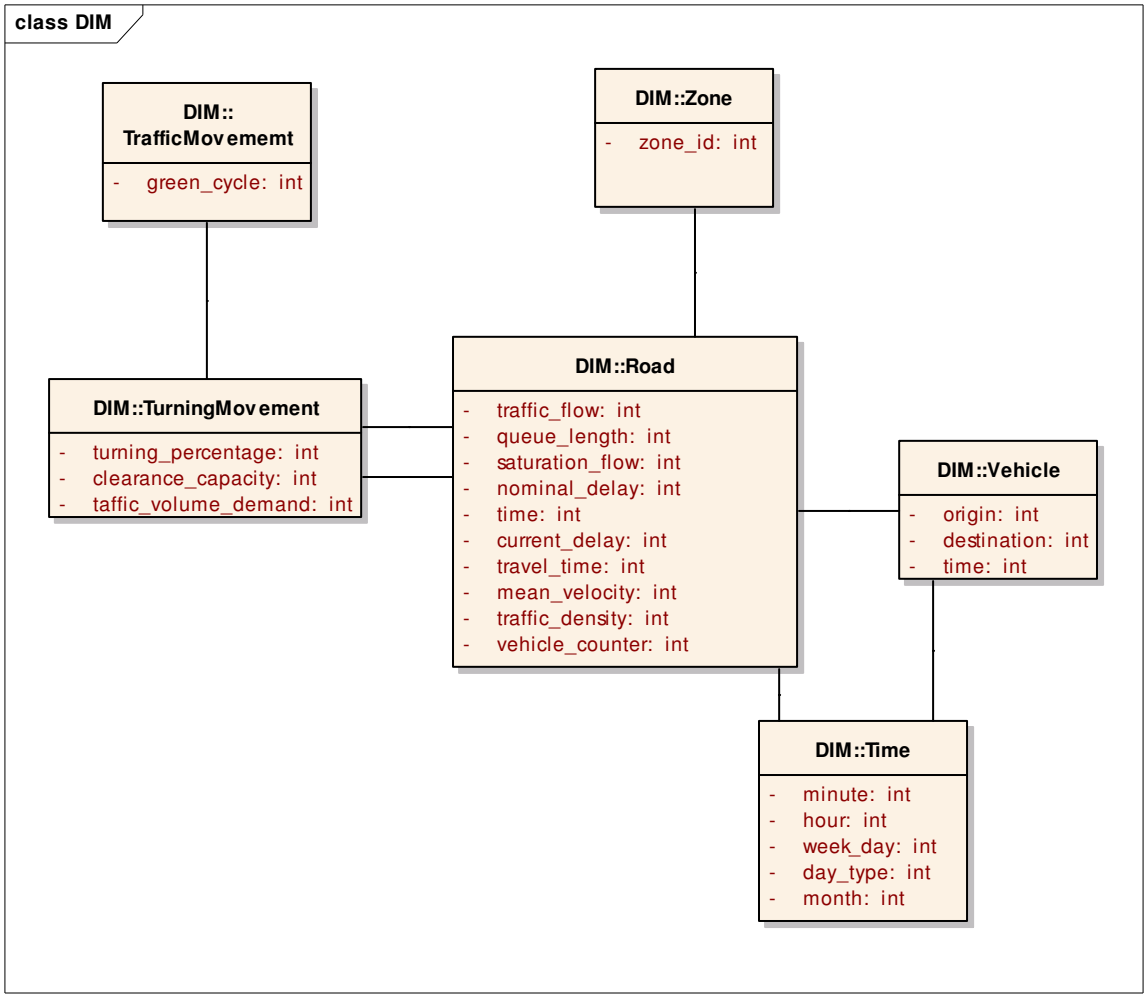

Figure 226: Information model of traffic control assessment application

In the domain information model the following information are represented:

The static road network representation.

The traffic value as coming from the vehicle (xFCD).

Traffic data from RSU, which are referred to the road.

Nominal status of the network.

Parameters of the traffic control model to be assessed.

7.14.4 Interaction model

The high level workflow of the traffic control assessment application is depicted in Figure 227. Data originated from the vehicle and the infrastructure are collected and integrated. The data is then archived and parameter integrated. The data collected and processed is then used for congestion warning and to allow analysis of the network.

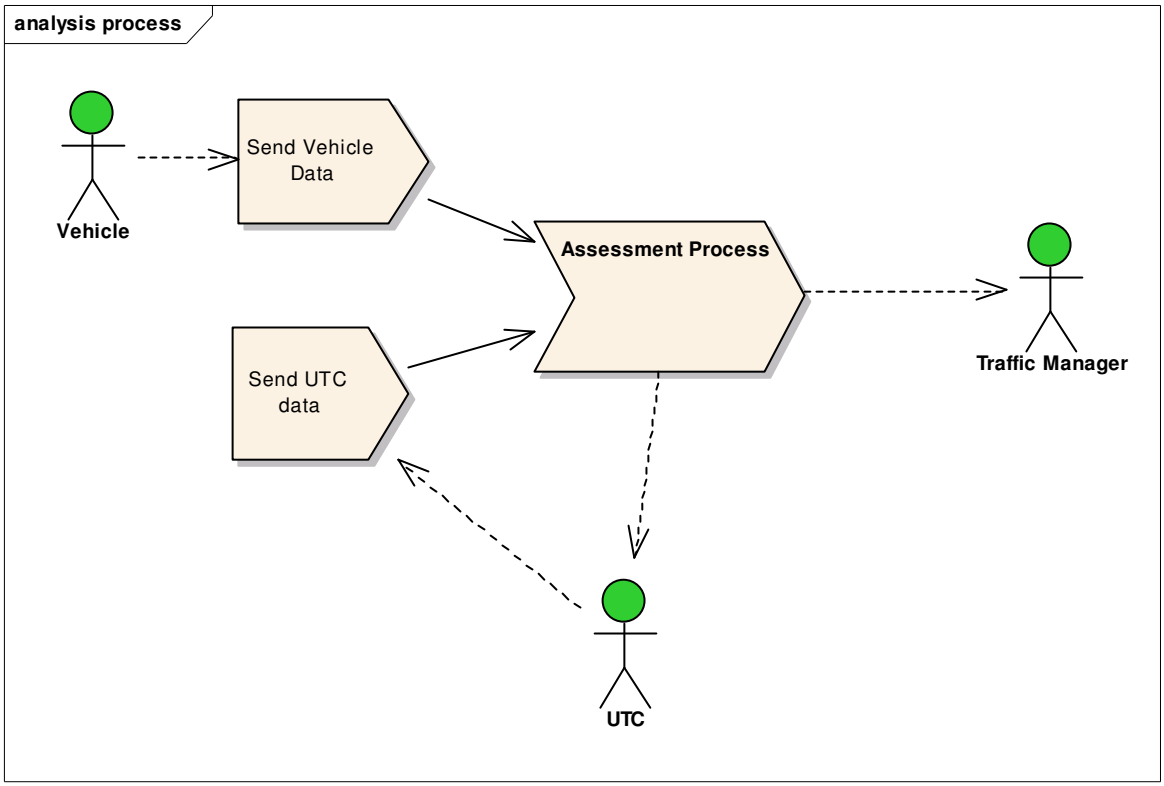

Figure 227: Reference service process traffic control assessment application

From the sequence diagram in Figure 228 it is possible to see how the information is exchanged. The UTC has the double role to provide local event data, provide current model parameters and to get new parameters of inconsistence generated by the traffic control assessment.

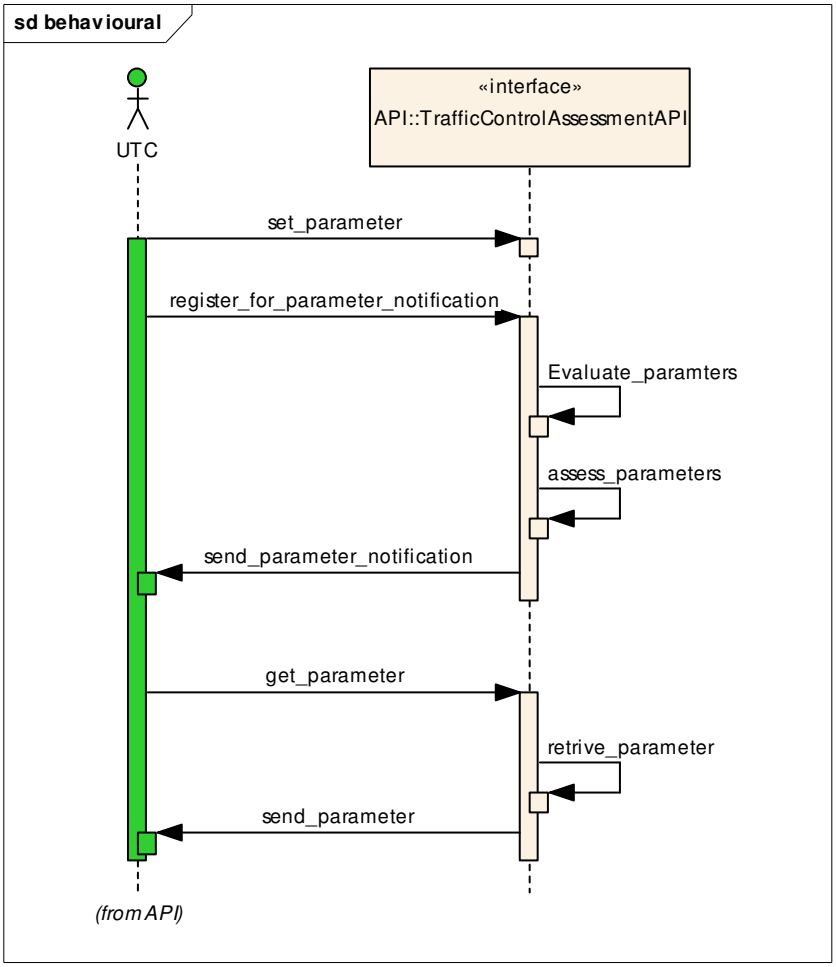

Figure 228: Interaction model traffic control assessment application

7.14.5 High level composite architecture

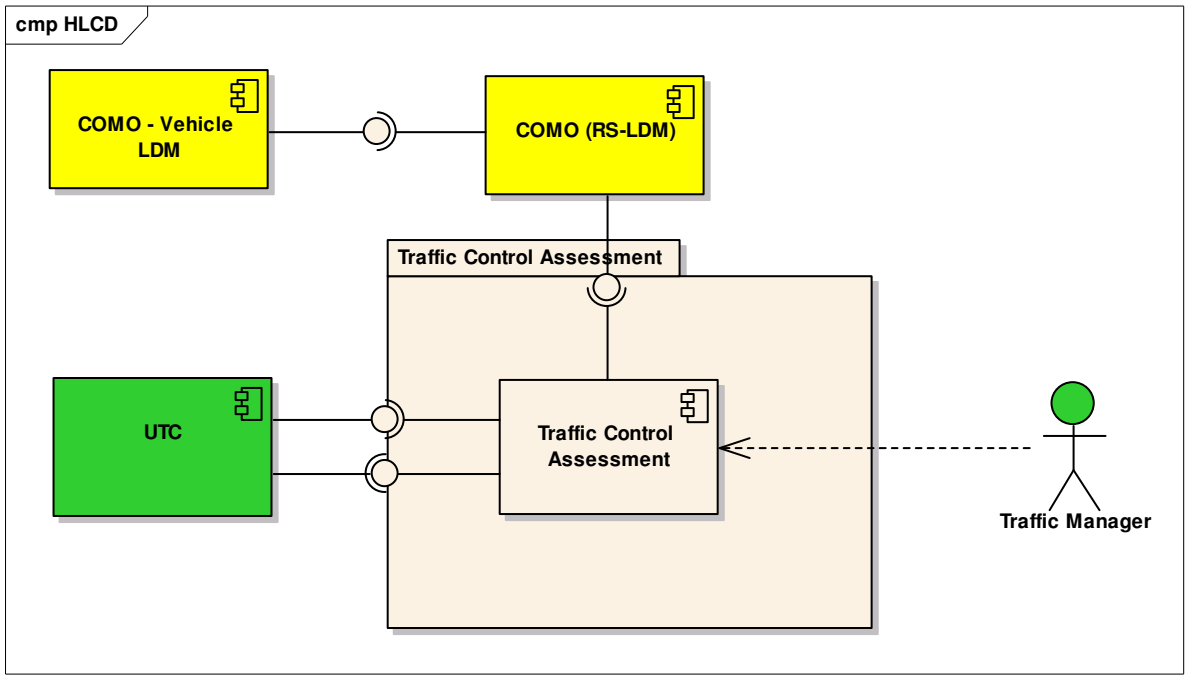

Figure 229: Component diagram of traffic control application

Traffic control assessment takes the traffic model, the local traffic network and the data from the vehicle and the RSU measurements. This information is then used to assess the traffic model parameters that are used by the traffic control unit. The tuned information is then sent to the traffic control unit.

7.14.6 Deployment model

The deployment diagram of the traffic control assessment is depicted in Figure 230.

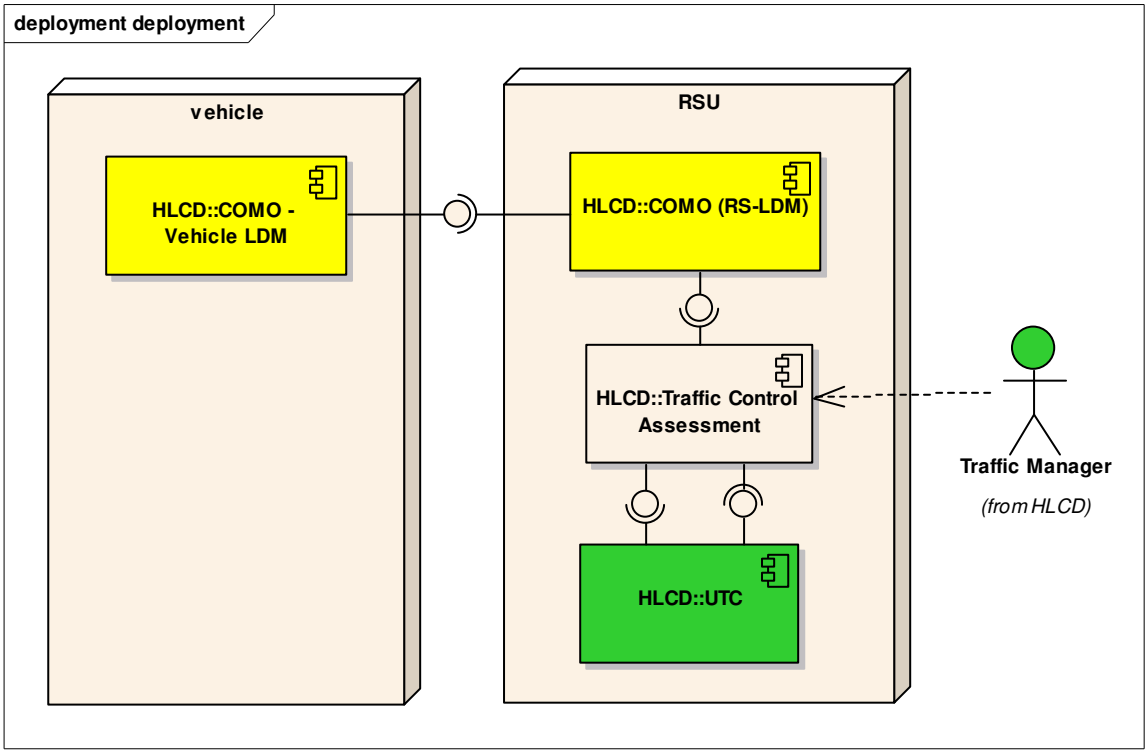

Figure 230: Traffic control assessment application deployment diagram

8 List of figures