UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

telecos BCN

Collaborative Perception Architecture in Smart Cities

Master Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

David Caules Pons

In partial fulfillment
of the requirements for the master in
ELECTRONIC ENGINEERING

Supervisor: M.Sc. Joao-Vitor Zacchi: Dr.Ing. Luis Javier De la Cruz Llopis
München, 12/04/2023

telecos BCN

Abstract

Autonomous Driving Systems have become a reality in our society. Everyday, progress is made to increase vehicles' autonomy to drive without restrictions in roads and cities. To achieve that, researchers are always seeking for new methods to ensure the safety of the vehicles. A promising strategy is to improve the quality of the collected perception data as it directly influences the overall performance of the autonomous system. However, despite the advances achieved in detection methods and algorithms, perception is currently physically restricted by the available on-board sensors and their line-of-sight.

To overcome this limitation, the autonomous system should not only capture on-board perception data, but also enhance it with data exchanged with other agents in the environment. This is known in research as Collective Perception, where mobile and stationary agents share object detection and sensor data inside an Intelligent Transport Systems network.

This master's thesis brings together a collection of ETSI standards with the goal of developing a well-defined architecture for future implementation of a Secure Collaborative Perception Network in the context of Smart Cities. The architecture has been designed using the open-source software Capella Arcadia following a Model Based Software Engineering methodology.

Keywords: Automated Driving Systems, Collective Perception, Intelligent Transport Systems, Model Based Systems Engineering

Acknowledgments

I would like to express my sincere gratitude to all those who have been part, directly or indirectly, of this master thesis project and all that concerns it.

Firstly, special mentions should be made of my supervisor, M.Sc. Joao-Vitor Zacchi, and Dr.Ing Núria Mata, not only for their constant help and dedication at any given moment, but also for offering me this great opportunity and the trust they have given to me from the start. My gratitude is also due to the great support and feedback received by Dr.Ing Luis Javier De la Cruz Llopis from Barcelna, who has always been there when needed as my supervisor.

Fraunhofer Institute for Cognitive Systems IKS gave me the opportunity to write this master thesis in the Cognitive Systems Engineering Department. Secondly, I want to thank the "Bayerisches Staatsministerium für Wirtschaft, Landesentwicklung und Energie" for the funding under the project "Unterstützung des thematischen Aufbaus des Instituts für Kognitive Systeme".

My infinite gratitude goes to my family, my father Toni, my mother Lena and my sister Neus, for always being the best advisors I could have ever wished for and for your unconditional support, not only through this master thesis, but through my whole life.

Finally, I would also like to thank my friends, who have made my academic journey through university unforgettable and made me the person I am today. Special appreciation to those who have accompanied me during the development of my thesis at IKS, Alex, Roger, Ken and Oliver.

Revision history and approval record

Revision	**Date**	**Purpose**
0	01/10/2022	Document creation
1	20/12/2022	Document revision
2	02/02/2023	Document revision
3	03/03/2023	Document revision
4	22/03/2023	Document revision
5	06/04/2023	Document revision
6	11/04/2023	Document revision

DOCUMENT DISTRIBUTION LIST

Name	
David Caules	
Joao-Vitor Zacchi	
Nuria Mata	
Luis Javier de la Cruz	

Written by:		Reviewed and approved by:	
Date	01/10/2022	Date	11/04/2023
Name	David Caules	Name	Joao-Vitor Zacchi
Position	Project Author	Position	Project Supervisor

Contents

List of Figures

List of Tables

1 Introduction

This master thesis has been developed at Fraunhofer Institute of Cognitive Systems, specifically in the department of Cognitive Systems Engineering, where one of the research focus is the modelling and research on Automated Driving Systems.

1.1 Motivation

Automated Driving Systems (ADS) can be divided into 4 high-level functions: Perception, Path-Planning, Localization and Vehicle-Dynamics Control. The importance of perception relies on the necessity to obtain data of the surroundings of the vehicle. Without this data it is impossible to perform any of the other functions and its quality directly influences the overall results of the system.

A lot of advances have been made in recent years into perception detection methods and algorithms. However, perception quality of current systems is constrained by the range of available on-board sensors and their line-of-sight [10].

In order to solve this limitation, research has been extended to Collaborative Perception (CP) where the sensors information is combined across multiple agents to improve the quality of the captured data. With the addition of CP, systems can obtain two important benefits: extension of a vehicles field-of-view, thus removing the line-of-sight sensitivity, and improving the confidence of observations as detections can be overlapped by two or more agents [10].

1.2 Goals

The main goal of this thesis is to define, implement and evaluate a Collaborative Perception Software Architecture for an Smart City environment. The developed software architecture is based on the guidelines proposed in the 'Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Specification of the Collective Perception Service, ETSI TS 103 324'[6] in addition to other ETSI from the ITS Collection.

The software architecture is developed using Arcadia Capella software, a powerful tool that allows the systems engineer to implement the architecture in different levels and views.

1.3 Scope

The focus of this master thesis is placed on the generation, transmission and reception of Collective Perception Messages, the managing of Object Detection information and methods to establish a secure communication. Meanwhile, The physical communication channel itself, the sensor data processing and object detections algorithms and processing times are out of the scope of this development.

1.4 Workplan

The project workplan is shown below. The starting date was at the end of September 2022. The first weeks consisted of learning about the topic to be developed by reading papers and some standards. Also, during the first month, it was necessary to learn the basics of Capella modelling software. Also, the report writing process started and a literature review was maintained through almost the entire master thesis in a constant search for new standards to add to the architecture. The architecture development is divided into three steps that were performed subsequently, although constant minor revisions were carried out through the whole development. Finally, the report ended with the completion of the results and conclusions part.

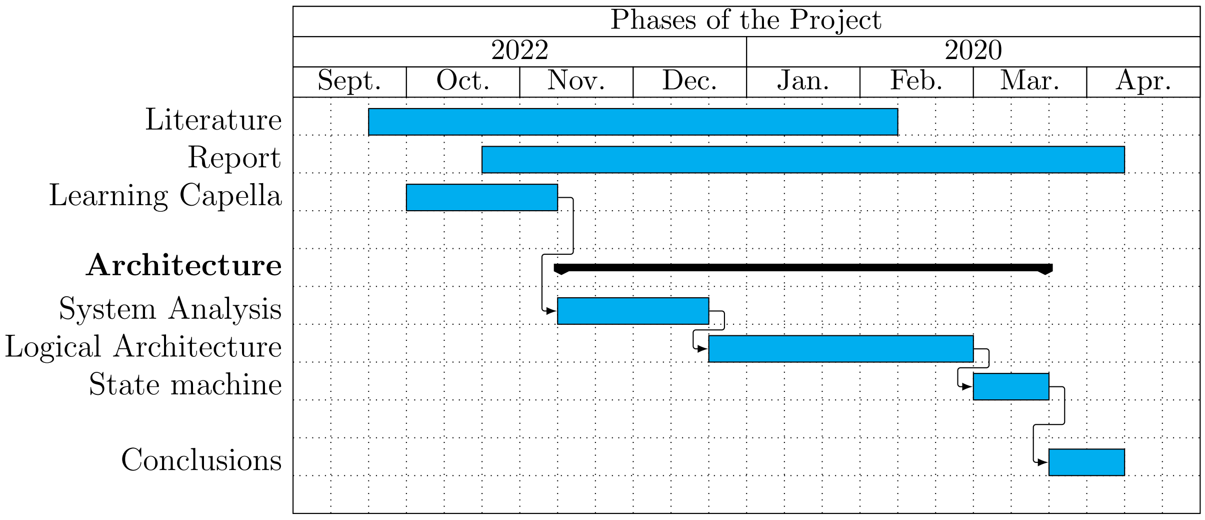


Figure 1: Gantt diagram of workplan

2 Terms, Symbols and Abbreviations

- **AA**: Authorization Authority
- **ADS**: Automated Driving Systems
- **AT**: Authorization Ticket
- **CAM**: Coordinated Awareness Message
- **CCMS**: Cooperative-ITS Credential Management System
- **CC**: Canonical Credentials
- **CP**: Collaborative Perception
- **CPS**: Collaborative Perception Service
- **CPM**: Collaborative Perception Message
- **CRL**: Certificate Revocation List
- **EA**: Enrolment Authority
- **EC**: Enrolment Credentials
- **ETSI**: European Telecommunications Standards Institute
- **ICT**: Information and Communication Technologies
- **ITS**: Intelligent Transport Systems
- **ITS-S**: Intelligent Transport Systems Station
- **MA**: Misbehaviour Authority
- **MBSE**: Model Based Systems Engineering
- **MD**: Misbehaviour Detection
- **MR**: Misbehaviour Report
- **LDM**: Local Dynamic Map
- **PKI**: Public Key Infrastructure
- **REC**: Replicable Elements Collection
- **RPL**: Replica
- **SC**: Smart City
- **SE**: Systems Engineering
- **SoS**: System of Systems
- **UML**: Unified Modeling Language
- **V2X**: Vehicle-to-Everything communication

3 State of the art

This chapter introduces the relevant work associate with this manuscript. First, it introduces the concept of Smart Cities alongside their proposed architecture and challenges. Second, Automated Driving Systems (ADS) Levels, main Capabilities and Challenges are explained. How these ADS can share data with each other trough V2X Communication is then explain. Next, the Intelligent Transport Systems (ITS) environment is introduced as a reference to establish an V2X network. The different levels and elements that compose an ITS architecture are described. Finally, an introduction is made to Systems Engineering and the different processes and tools that are used, to introduce the development part of this thesis.

3.1 Smart Cities

3.1.1 Definition

There is no universal definition for a Smart City (SC), as there is no single model to build one, every city in the world is unique. Different authors have proposed definitions for the Smart City concept. Some examples are:

Kulkarni et al. *"The smart city refers to a local entity, i.e., a distinct city, region or small locality which takes a holistic approach to employing information technologies with real-time analysis that encourages sustainable economic development."*[11]. From this definition three main topics can be extracted: locality, information technologies and sustainability.

Nelson et al. *"The smart city is a territory with a high capacity for learning innovations, built on the creativity of its residents, their knowledge development, and their digital infrastructure for communication and knowledge management."*[12]. In this second definition the concepts of a locality and technology can be found again, although it focuses more on the knowledge development.

These main concepts keep appearing on different authors definitions, with two final examples being:

Wilson et al. *"Smart city has high productivity as they have relatively higher educated people, knowledge-intensive jobs, output-oriented planning systems, creative activities and sustainability-oriented initiatives."*[13]

Xiong et al. *"The applications of Information and communication technology with their effects on human capital, education, social and relational capital and environmental issues are often indicated by the residents of the smart city."*[14].

Despite the absence of an unified concept, from a technical perspective, Smart Cities are areas where Information and Communication Technologies (ICT) are widely used in infrastructures and services. SC share common attributes, such as sustainability or knowledge development. By putting all these common attributes together authors have defined a set of key characteristics over which smart cities are build: Sustainability, Quality of life, Smartness and Urbanization[15].

3.1.2 Architecture

Various works introduce Smart City software architectures to manage the complexity and ease its implementation. Table 1 shows a variety of proposals with different goals and applications. However, due to the high variability in specific needs, a generic implementation approach does not exist.

Architecture	**Sensing Layer**	**Network Layer**	**Data Management Layer**	**Application Layer**
Zanellaa et al.[11]	Deployed a sensor network	A communication approach using constrained application protocol (CoAP)	Applied an averaging filter for 10 reading to process data	Proposed HHTP-based interface for each IoT node
Silva et al.[12]	Proposed deploying sensor network	Occupied various transmission mediums to gather data	Proposed a Kalman filter optimization based Big Data analytic approach	Proposed a partitioned application layer to optimize event execution
Al-Hader et al.[13]	Proposed a geographic information system (GIS)	-	Proposed application, business and information management approach	-
Anthopoulos and Fitsilis[14]	Represented public and private data creation and storage	Incorporated various transmission mediums	Introduced enterprise architecture, policies and rules	Provide information and services without information replication
Lugaric et al.[14]	Data acquisition via physical grid	Agent based modelling for communication infrastructure	Agent based modelling for data exchange and decision-making	-

Table 1: Some Smart City Architecture proposals

Although the specific implementation has not been possible to generalize, authors have reached a consensus regarding the overall conceptual architectural and main elements distribution. In Figure 2 it can be observed how the different functions from a SC are organized and divided into different layers.

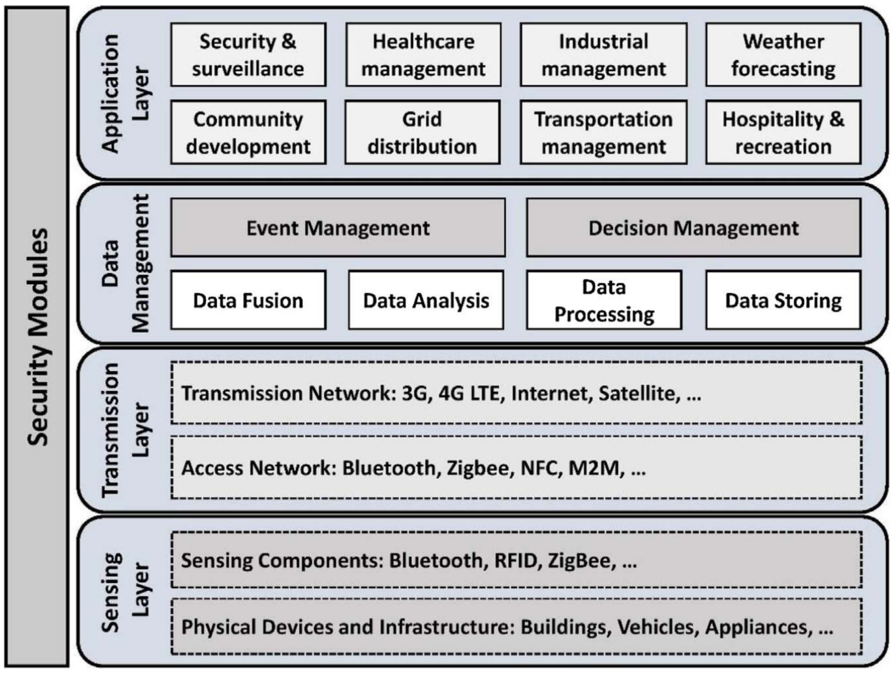


Figure 2: Layered architecture of generic smart city[1]

- **Sensing Layer:** Sensing layer provides an information collection interface, which is of vital importance for the management as it will be the foundation for all future decision-making. Additionally, because the information is so varied, obtaining data is not a simple task.
- **Transmission Layer:** Transport layer is the assembly of different correspondence organization that compose the primary support of the smart city architecture to connect data sources with management stations.
- **Data Management Layer:** To reach the desired decision, all the received data must first be processed, then analyzed. It is possible to carry out a variety of tasks, including combination, information preparation, stockpiling, occasion, decision management, etc.
- **Application Layer:** Citizens only perceive performance based on the application output since they are unaware of the data management layer. The application layer handles and stores information in a variety of ways to enhance city operation. Additionally, information exchange between various applications is promising for intelligent urban planning.

Applications in Smart Cities can be developed to improve a diverse range of city components: education, healthcare, energy consumption, transportation, etc. This thesis focus on the transportation part of the SC scheme. One of the best ways to obtain a better transportation network is by collectively using information and communication technologies instead of the traditional individual transportation.

This information can then be used for example to provide passengers with information about road congestion, alternative routes and modes of transportation, etc.[15]. Vehicular ad hoc networks (VANET) have raised in attention with the integration of Intelligent Transportation Systems (ITS) which use vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication to share real-time data.

The integration of smart transportation systems in smart cities upgrades the operational efficiency of the city and at the same time optimizes time, cost, reliability and safety of the city mobility[1].

3.1.3 Challenges

Although the Smart City concept is already widely accepted and there has been seen some real implementation, it still faces challenges in certain areas that are crucial for further development. Authors in [1] and [15] go through a review of the main challenges for the future of smart cities with respect to: cost, security, heterogeneity, data collection and analysis. A summary of each category is described below:

Cost To establish a real smart city implementation, design and maintenance cost are one of the major problems to be tackled. Design cost is the financial capital needed for deploying the smart city initial infrastructure, while operational costs are the capital needed for daily city operations and maintenance tasks. Low initial design costs augment the probability of real-world implementation of smart cities and minimal operational costs are demanded to assure sustainability on the long run. For example, reducing Carbon Footprint by having an efficient management of resources and usage of renewable energies or having great Failures Management against natural disasters, can be two great functionalities to be added to a SC, but they come with a cost demand. The real challenge resides on finding the trade-off between functionalities and performance with costs.

Security Smart cities are susceptible to harmful malware attacks due to their reliance on data. This vulnerability poses a serious threat to the city's management systems, as these systems are responsible for coordinating critical functions that are vital to the city's smooth operation. Thus, it is crucial for smart cities to implement strong security measures to safeguard against such attacks and mitigate their potential impact.

In addition to the risks of malware attacks, smart cities are also vulnerable to data breaches that compromise the privacy of citizens. This is particularly concerning given the constant collection of sensitive citizen information. Such breaches can occur through attacks such as sidewalk attacks, cross-site scripting, and other similar threats. Sidewalk attacks occur when a hacker intercepts data as it is being transmitted from a device, while cross-site scripting attacks take advantage of vulnerabilities in websites to access and manipulate sensitive information. To ensure the confidentiality, integrity, and privacy of collected data, it is necessary to have robust security measures in place, which can be expensive to implement.

Furthermore, the usage of a wide range of devices such as computers, smartphones, and other smart devices by residents to access smart city services can increase the risk of data interception, which is another pressing concern that must be addressed to maintain the privacy of sensitive data.

Heterogeneity Integrating the multitude of sensors, devices, and appliances from various vendors in a smart city is a daunting task due to their heterogeneity. The presence of platform incompatibilities further complicates the process of integrating and interoperating these elements at the application layer, preventing the full realization of the potential of smart cities. Additionally, integrating such heterogeneous critical systems poses significant challenges from a safety standpoint, as protecting such data becomes an even bigger concern.

Data Collection and Analysis The exponential growth of data generated from the numerous devices in smart cities presents a significant challenge for efficient transfer, storage, retrieval, and analysis. The management of large volumes of data is critical to the smooth functioning of smart cities. As a result, smart cities are continually searching for innovative ways to handle and analyze big data to ensure uninterrupted and seamless operations.

3.2 Automated Driving Systems

Automated Driving Systems (ADS), also known as self-driving or autonomous vehicles, have gained significant attention in recent years as a promising technology for improving transportation safety and efficiency. In this chapter, an overview of the current state of the art on ADS is provided, as well as the challenges and opportunities that lie ahead in the development of these systems.

3.2.1 SAE Automated Driving Levels

As described by the Society of Automotive Engineers (SAE), transferring total control from humans to machines is a step-by-step process on a scale of 0 to 5 levels. Level 0 means no automation and level 5 means full-time performance by an automated driving system under all road and environmental conditions. SAE J3016 [16] defines the multiple levels of driving automation. The standard provides descriptive and broad information about this evolution but does not specify strict requirements.

SAE classifications are designed to clarify the role of a human driver, if any, during vehicle operation. An environmental monitoring agent is the first discriminant condition. For levels 0-2 of automation, a human driver monitors the environment; while for levels 3-5 of automation, the vehicle monitors the environment.

Another discriminant criterion is dynamic driving task (DDT) fallback mechanisms. Intelligent driving automation systems (levels 4-5) embed the responsibility for automation fallback constrained or not by operational domains, while for low levels of automation (levels 0-3) a human driver is fully responsible. Figure 3 shows the remaining classification factors used to define each level [16], [2].

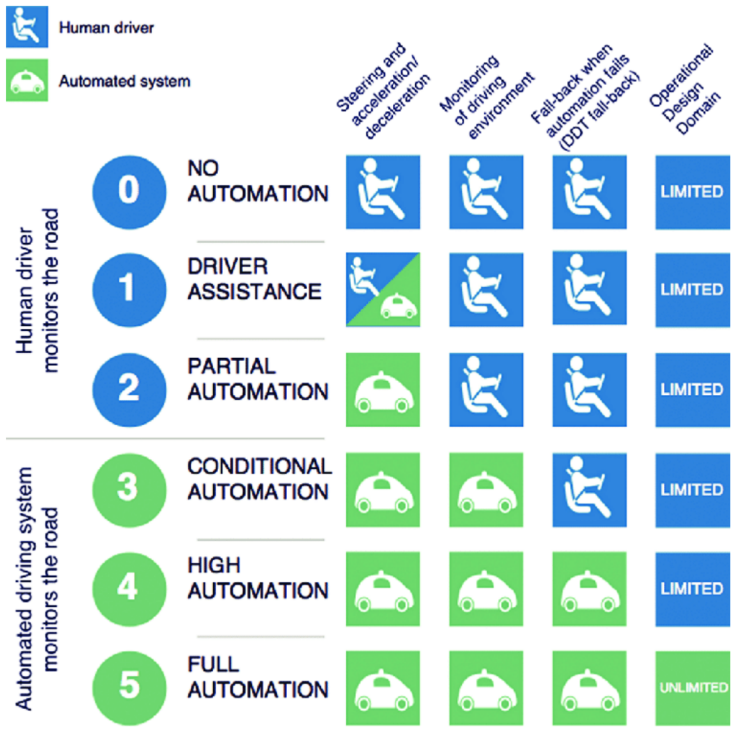

Human driver
Automated system
Steering and acceleration/ deceleration
Monitoring of driving environment
Fail-back when automation fails (DDT fall-back)
Operational Design Domain
Human driver monitors the road
Automated driving system monitors the road
0 NO AUTOMATION
1 DRIVER ASSISTANCE
2 PARTIAL AUTOMATION
3 CONDITIONAL AUTOMATION
4 HIGH AUTOMATION
5 FULL AUTOMATION
LIMITED
UNLIMITED

Figure 3: Automated driving levels[2]

3.2.2 ISO/TR 4804

The "ISO/TR 4808 Road vehicles — safety and cybersecurity for automated driving systems — design, verification and validation" [3] outlines a framework for the development, verification, validation, production, and operation of automated driving systems with a focus on safety and cybersecurity. In particular, the document focuses on verification and validation methods for automated driving systems at levels 3 and 4 of the SAE J3016[16] classification, although it may also be applicable to lower levels of automation.

The technical report in question provides a general strategy for addressing the risks associated with automated vehicles. It is intended to be used as a starting point for the development of safe automated driving systems, but it does not provide a comprehensive and complete solution for ensuring the safety of such systems. Instead, it offers a general framework that can be adapted and customized to the specific needs and requirements of individual automated driving systems. It is important to note that this technical report should be used in conjunction with other relevant standards and guidelines, in order to ensure the safety and reliability of automated vehicle systems.

Capabilities of automated driving An automated driving system has a basic set of system properties that are specified as capabilities. These are divided into two categories: fail-safe and fail-degraded. Fail-safe capabilities are those that provide customer value and enable the system to operate, but can also bring the system to a minimal risk condition in the event of a failure. These capabilities can be discontinued if the safety relevance of their unavailability is low enough or if it is covered by other fail-degraded capabilities. Fail-degraded capabilities, on the other hand, are those that do not provide customer value but are necessary to ensure the safety of the system. These capabilities cannot be discontinued and must be maintained in order to prevent the system from entering a hazardous state. Fail-degraded capabilities are typically used to monitor system's performance and detect potential failures, as well as to trigger appropriate responses to ensure the safety of the system and its occupants[3].

The operation of automated driving systems can be understood using the classic sense-plan-act design paradigm (Figure 4) from robotics and automation literature. In this model, the components of the system are divided into three main categories: sensing and perception, planning and control, and actuation and stability. These categories provide a general, implementation-independent view of the system and its functions and together they enable the system to operate autonomously and make driving decisions.

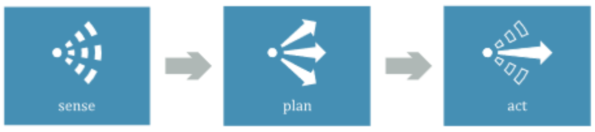


Figure 4: Sense-Plan-Act design paradigm[3]

Based on the allocation of capabilities to the basic functions for sense – plan – act, it is possible to allocate requirements for elements that the automated vehicle is reasonably safe as depicted in Figure 41. In this master thesis, the focus is placed on two capabilities: *FS_2: Perceive relevant static and dynamic objects* and *FS_6 Communicate and interact with other road users*. A description of both is located in the Appendice A.

3.2.3 Safety Challenges

A multidisciplinary approach across all levels of the functional hierarchy is necessary to ensure the safety of fully autonomous vehicles, including hardware fault tolerance, resilient machine learning, collaboration with humans operating conventional vehicles, system validation for use in highly unstructured environments, and appropriate regulatory approaches. Authors in [17] explore the different range of safety challenges and concerns that need to be faced. Safety Engineering and Social Acceptance are explained in the next paragraphs. The remaining challenges are explained in Appendice B.

Safety Engineering Assuming the existence of small-scale deployment of fully autonomous Level 4 vehicles on the road, the challenge becomes managing failures that may be infrequent for any single vehicle, but will happen often when deploying at a big scale. The current accepted practice for vehicle computer-based system safety, which relies on the driver to be responsible for vehicle safety, needs to be modified as a fully autonomous vehicle eliminates the human driver's responsibility entirely. Another safety certification concern is validating the use of self-adaptive system behavior by autonomous vehicles, which may lead to different behaviors during operation than was displayed during testing and certification.

One method for ensuring safety in highly autonomous systems is to switch control back to a human driver in the event of an equipment failure, allowing them to take over driving responsibility. However, if the human is not paying attention, there needs to be a fail-operational autonomy capability in place to maintain control until the human can take over. Cars can achieve a safe state quickly, so a strategy is to switch to a short "safing mission" mode when a critical component fails, allowing the car to pull over to the side of the road without the need for complex driving autonomy. This "safing mission" has less strict reliability and redundancy requirements and potentially reduce overall system cost and complexity, while still keeping the vehicle safe. By designing a safe shutdown mission capability, the safety requirements on primary vehicle autonomy can also be relaxed.

Social Acceptance The process of gaining social acceptance for autonomous vehicles will undoubtedly be complex. While a primary incentive for adoption is the expectation that autonomous vehicles will be safer drivers than humans, it is unrealistic to assume that this will mean zero accidents, especially in the early stages of implementation. Some cases will be straightforward, such as when avoiding a collision is physically impossible, such as when a tree falls on top of a car during a storm. However, determining the standard for autonomous safety will be challenging, such as whether it should be better than an excellent human driver or just a typical one, and how a typical driver should be characterized. Particularly tricky situations will arise when an ordinary human driver could have avoided an accident, but the autonomous vehicle did not, leading to questions about the vehicle's liability in such cases.

3.3 Vehicle to Everything (V2X) Communication

Communication between vehicles and other objects (V2X) has the potential to significantly advance vehicle safety technologies. It allows V2X equipped vehicles to exchange their telemetry data to raise awareness, especially in NLoS (Non Line-of-Sight) circumstances.

A message known as the Basic Safety Message (BSM), also known as the Coordinated Awareness Message (CAM), is sent to accomplish this. Although they are defined according to two distinct standards, BSM by the Society of Automotive Engineers (SAE)J2735[18] and CAM by the European Telecommunications Standards Institute (ETSI) European Standard (EN) 302 637-2[19], both messages have the identical content. From now on, we will only refer to both as "CAM Service".

Through the usage of CAM, connected vehicles can share their location and kinematic state (velocity and acceleration) to other surrounding vehicles. Then, this extended information can be used in safety applications, such as Intersection Collision Warning [20].

Nevertheless, not every item on the road may have V2X capability, e.g., non-V2X vehicles, pedestrians, obstacles, animals. As a result, a brand-new service called Collective Perception Service (CPS), also named as Collaborative or Cooperative Perception, has been developed to overcome this limitation, enabling the V2X vehicles to share observations of non-connected items. The Collective Perception Message (CPM), produced and consumed by participating vehicles, is intended to supplement the CAM service[21].

CPS is a method based on improving the perception quality of autonomous systems by combining sensor information from multiple agents, such as multiple autonomous vehicles or a combination of autonomous vehicles and infrastructure. This allows the system to overcome the limitations of individual on-board sensors and improve the overall perception range and accuracy. In addition, CP not only extends the field-of-view of individual vehicles, but also can improve the confidence of observations within an area that is overlapped by the perception range of two or more connected agents. Overall, cooperative perception has the potential to significantly improve the perception quality of autonomous systems, enabling them to make more accurate and reliable decisions in complex and dynamic environments[10].

The standard ETSI 103 324 Collective Perception Service [6] is, at the moment of writing this report, under development with its publication date set for May the 5th of 2023.

3.4 Intelligent Transport Systems

In ETSI standards the V2X forms part of the Intelligent Transportation System (ITS) Environment, which has already a well developed collection of standards focusing on a diverse range of subtopics.

In this environment there are different ITS-Stations (ITS-S) that will be able to exchange messages and information inside a defined communication network. This ITS-S can be from a diverse origin, from moving vehicles, to roadside stations or pedestrians.

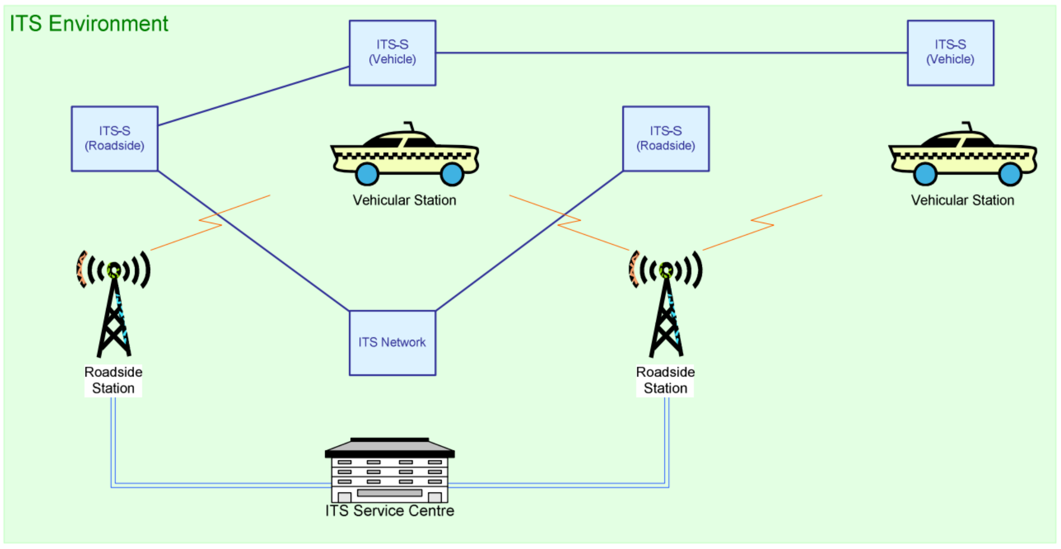


Figure 5: ITS Environment[4]

ETSI Standards have a well defined architecture for ITS-S. As it can be observed in Figure 6, the architecture is distributed in different layers according to the different functionalities they contain:

- Application Layer: provides ITS services using the information available in the other layers. Three classes of applications have been distinguished at this level: Road

Safety, Traffic Efficiency and Other Applications.

- Facilities Layer: provides generic support facilities to applications. The nature of the different facilities can be diverse: application support, for example time management, information support, like detection's database, or communications support like messages generation.
- Network and Transport Layer: one single layer for network and transport functions. It has the possibility to incorporate Multi-protocol options.
- Access Layer: constitutes the physical level of the communication channel, the technology that is used to send and receive messages (e.g. 5G, Ethernet, etc.).
- Management: in charge of managing internal communications inside the ITS station.
- Security: provides security services to the management entity. Can also be considered as a specific part of the management entity itself.

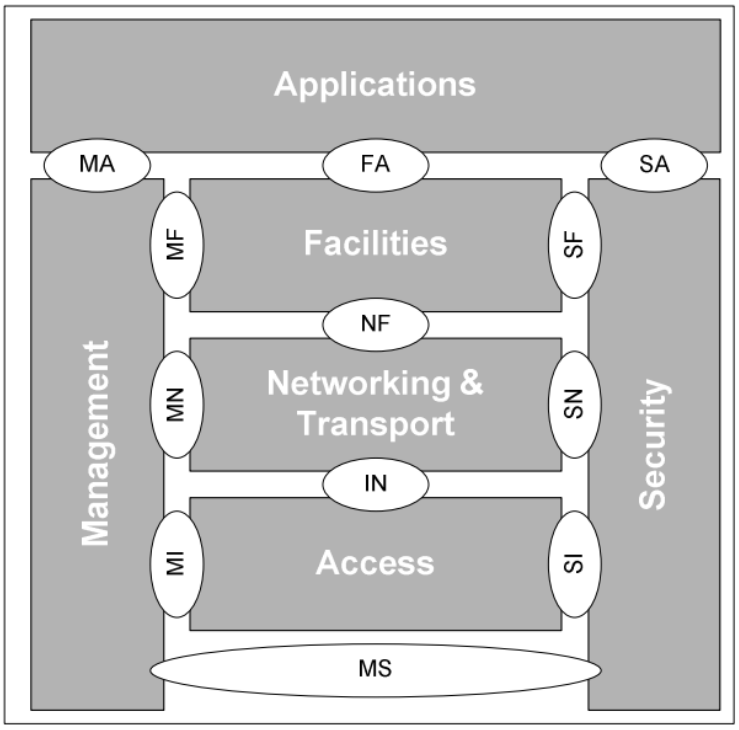


Figure 6: ITS Station reference architecture[5]

3.4.1 CP Service

The CP Service is a facility that operates the CPM protocol and provides two functions: sending and receiving Collective Perception Messages . It uses the services provided by the protocol entities of the ITS (Intelligent Transportation System) networking and transport layer to transmit the CPM.

CPMs are used in the ITS network to share information about the perceived environment of an ITS subsystem, including the presence of road users, objects, and free spaces on the road. These messages are exchanged between ITS-Ss (Intelligent Transportation System - Systems) to help them understand and navigate their environment, including identifying areas of the road that are free from road users and collision-relevant objects.

The general structure of a CPM is illustrated in Figure 7. Different containers constitute the whole CP Message. The main containers are discussed in the following paragraphs.

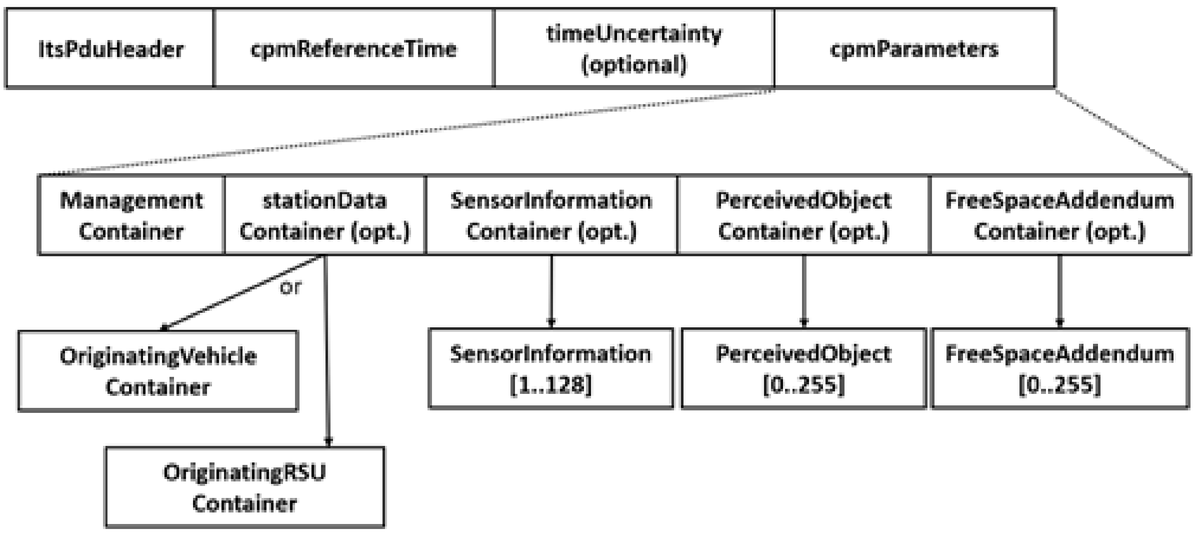

ItsPduHeader
cpmReferenceTime
timeUncertainty (optional)
cpmParameters
Management Container
stationData Container (opt.)
SensorInformation Container (opt.)
PerceivedObject Container (opt.)
FreeSpaceAddendum Container (opt.)
or
OriginatingVehicle Container
OriginatingRSU Container
SensorInformation [1..128]
PerceivedObject [0..255]
FreeSpaceAddendum [0..255]

Figure 7: General Structure of a CPM [6]

ITS PDU Header The ITS PDU header is a common header that includes the information of the protocol version, the message type and the ITS-S ID of the originating ITS-S. The Management Container offers details about the ITS-S Type and the Reference Position of the ITS-S, regardless of whatever type of ITS-S distributes a CPM, which can be either a moving ITS-S, e.g. a vehicle, or a stationary ITS-S, like a roadside unit.

Station Data Container If a CPM is generated by a vehicle-based ITS-S, then the Station Data Container must contain the originating vehicle information object. This container shall hold the dynamic information pertaining to the ITS-S originating the message.

On the other hand, if the CPM is generated by a roadside ITS-S, then the Originating Roadside ITS-S Container, containing the originating roadside information object, may be present. If present, it should provide references to identification numbers supplied by the MAP Message disseminated by the same roadside ITS-S.

Sensor Information Container The Sensor Information Container provides information about the sensory capabilities available to an ITS subsystem. Different container specifications are available to encode the properties of a sensor, depending on the ITS-S type at origin. If the time elapsed since the last CPM that included a sensor information is equal to or greater than T_AddSensorInformation, which is set to 1000ms, then a new SensorInformationContainer must be included in the CPM. It is worth noting that Sensor Information Containers are attached at a lower frequency than other containers, so it is not mandatory to attach them in every message.

Perceived Object Container The information about objects should be added to the Perceived Object Container by including a PerceivedObject DF. The inclusion management of the objects depends on the profile configuration. If the profile UseObjectInclusionRules is set to "false," all or some of the known objects will be included in the Perceived Object Container. However, if the profile is set to "true," the inclusion rules will apply differently for Type-A and Type-B objects.

To determine which objects to include in a CPM generation event, the transmitting ITS-S selects objects from the object list that meet certain conditions. These conditions vary depending on the assigned object class, which is either Type-A, objects with a subclass from profile pedestrian, bicyclist and light vehicle or animal, or Type-B, which are objects of any other class.

For Type-A objects, the transmitting ITS-S will include the object in the CPM if:

- It was first detected by the perception system after the last CPM generation event
- There is at least one object of Type-A in the object list that has not been included in a CPM in the past 500ms, in which case all Type-A objects should be included in the currently generated CPM.

For Type-B objects, the transmitting ITS-S will include the object in the CPM if:

- It was first detected by the perception system after the last CPM generation event
- The distance between the current position of the reference point of the object and the position of the same point in the last CPM exceeds 4m
- The difference between the current estimated ground speed of the reference point and the estimated absolute speed of the reference point in the last CPM exceeds 0.5m/s
- The orientation of the estimated object's ground velocity at its reference point has changed by at least 4 degrees since the last inclusion of the object in a CPM
- The time elapsed since the last inclusion of the object in a CPM exceeds T_GenCpmMax

To reduce the number of generated messages, objects belonging to Type-B that are to be included in the next generation event may be included in the currently generated CPM. This may be achieved by predicting objects that are not selected for transmission in the current message to the next CP message generation event. The predicted objects that would then need to be included in a CPM in the next generation event may also be selected for inclusion in the currently generated CPM.

Free Space Addendrum The SensorInformationContainer in a CPM can contain information about free space and objects. Figure 8 illustrates the physical definition of the free space.

If the free space area computed on the receiver side using the simple tracing approach does not reflect the detected free space of the ITS sub-system generating the CPM, the FreeSpaceAddendumContainer together with the corresponding FreeSpaceAddendum DFs shall be added.

In cases where there is static information, such as a permanently shadowed region, the FreeSpaceAddendumContainer shall be added whenever the SensorInformationContainer is added to the currently generated CPM. Additionally, a CPM generated as part of a generation event may include additional information about monitored free space areas known to the transmitting ITS-S by adding a FreeSpaceAddendum DF to the FreeSpaceAddendumContainer.

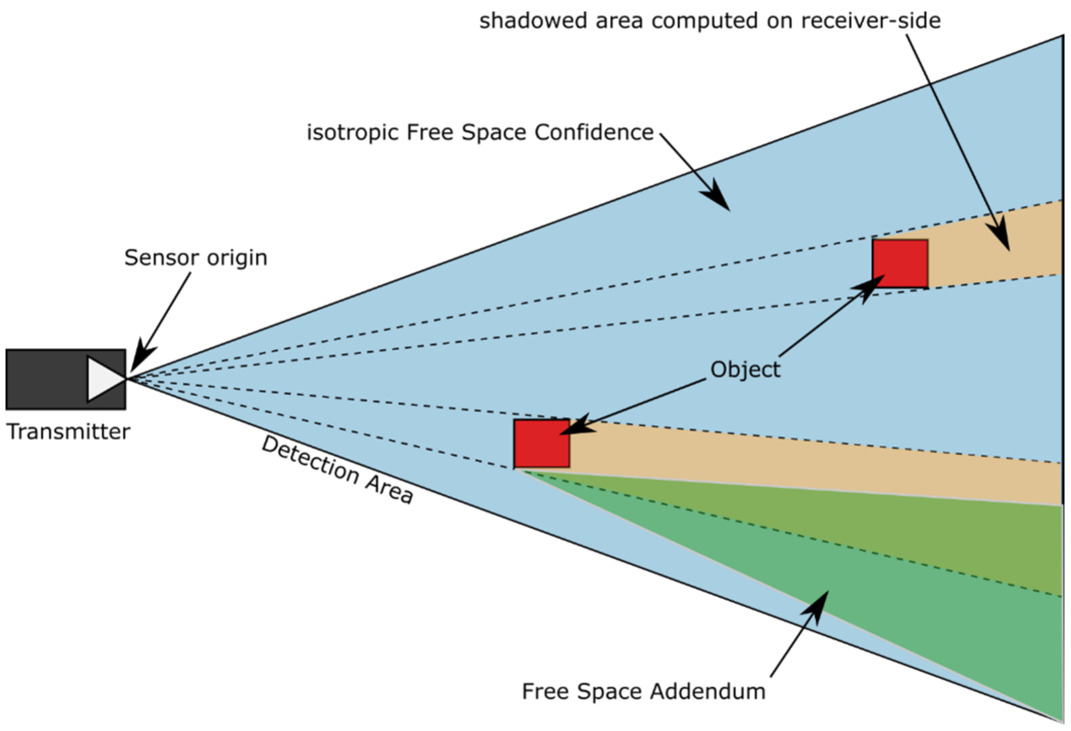

shadowed area computed on receiver-side
isotropic Free Space Confidence
Sensor origin
Object
Transmitter
Detection Area
Free Space Addendum

Figure 8: Free Space definition [6]

3.4.2 Local Dynamic Map

A Local Dynamic Map (LDM) is part of the facilities of collaborative ITS. ITS applications require data on moving items, such as adjacent automobiles, as well as fixed objects, such as traffic road signs. Therefore, the LDM is a conceptual data store that is housed inside an ITS-S and contains data that is pertinent to the functioning of ITS applications and matters of connected road safety and traffic efficiency. Vehicles, infrastructure components, traffic centers, personal ITS stations, on-board sensors, and other devices can all provide data sources. The LDM provides tools for granting safe access to the data it manages.

The access from Data Consumers to the data stored in the LDM is done through either a publish/subscription for the applications or functions that need a data for a periodic event or a query interface for a single request.

3.4.3 Position and Time Management

The Basic System Application specification (BSA) ETSI TR 102 638 [4] requires the exchange of position and time information among ITS-Ss. This is facilitated by all CAM, DEN and CPM, which are dependent on position and time information. To enable the absolute knowledge of position and time for road safety ITS applications, ITS-Ss may be equipped with a Global Navigation Satellite System (GNSS). The accuracy, integrity, and reliability of position and time references may vary based on the ITS application requirements. The PoTi entity in the facility layer manages position and time information for use by ITS applications and services. Various augmentation methods may be applied to improve the accuracy of position and time, and PoTi may support these augmentation services by broadcasting augmentation data through message services [22].

3.4.4 Basic Transportation Protocol

The Basic Transport Protocol (BTP) is a connection-less transport service that operates within the ITS ad hoc network. Its primary function is to multiplex messages from various processes in the ITS facilities layer, such as CAM or CPM, and enable transmission through different protocols.

Message multiplexing/demultiplexing is accomplished using 16-bit addresses called ports, which function as communication endpoints that identify the source or destination protocol entity of an ITS station. This is similar to the two-stage packet transport system in the IP protocol suite. BTP is a lightweight protocol with minimal processing requirements, and its design assumes that its users are either tolerant of unreliable packet transport or have reliable communication mechanisms built into their protocols [23].

The different assigned port numbers can be found in ETSI 103 248 [24]. For instance, 2001 corresponds to CAM, 2002 to DENM and 2009 to CPM. The complete list of BTP Port numbers can be found in the Appendice E.

3.4.5 Secure Communication

Security issues raise with the implementation of an ITS network as a malware-agent can connect to the general network and send misleading information regarding safety-critical systems. For example, imagine an scenario where one false agent sends a CPM with information regarding the state of a traffic-light on a crossing. If this information was taken as true without scrutinising the consequences can be a fatal crash between ITS vehicles.

One of the possible solutions is to implement a Public Key Infrastructure (PKI), also known as a Cooperative-ITS Credential Management System (CCMS), which serves as a certificate management system that facilitates the secure distribution, use, and revocation of certificates to ITS stations (ITS-S). Revocation of trust credentials may take place under different situations, for example, the CCMS determines to remove a malicious ITS station from the network after spotting it or the certificates granted to an ITS station are withdrawn when the station reaches its ”ITS-S end of life” [9].

There are two types of certificates that an ITS-S must request before being able to make full use of the ITS applications and services: Enrolment Certificate (EC) and Authorization Tickets (AT). The Enrolment Certificate are a long term credential, issued by the Enrolment Authority (EA), that grants permission to the ITS-S to be part of the ITS Network as a certified and identified agent. However, once enrolled, if the ITS-S wants to make use of one specific service, it must request Authorization Tickets to the Authorization Authority (AA). These AT are single use and must be added for every execution of that particular service, for example, every time a CPM wants to be added to the communication network, an AT must be used [25]. A diagram of this exchanges can be seen in Figure 10.

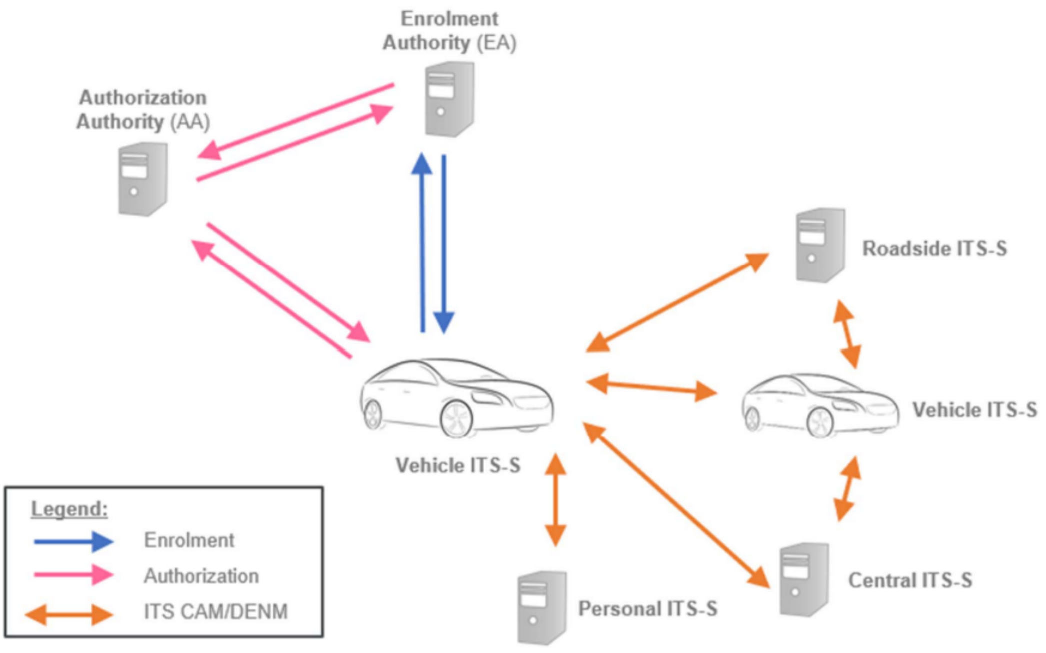


Figure 9: ITS Communication PKI Reference Scenario [6]

3.4.6 Misbehaviour Detection

Another layer of security can be added to the previous PKI scenario by not trusting completely the agents that have been granted a certificate and constantly checking that the data that is being shared does not have any errors, thus possibly detecting malfunctioning or malware agents. This process is known as Misbehaviour Detection and Reporting.

Misbehaviour detection and reporting is a main issue and does not have a trivial solution for various reasons, such as that denigration of benign ITS stations cannot be ignored, having the risk of false positive. In addition, having a network connection to the PKI backend server is necessary for misbehavior detection. It cannot be taken for granted that there is always an open line of communication. As there are no real-time requirements for the transmission of "Misbehaviour Reports" (MRs), ITS stations may store information on observed inappropriate behaviors or suspicious messages and submit it to the PKI server, also known as the "Misbehavior Authority" (MA), when a communication link is available.

Despite all challenges, misbehaviour detection and reporting should be considered from the start of the design of ITS Stations [9].

Different detection approaches are proposed in state of the art published works. Safety services within C-ITS rely on the participation of every communication node in the network, each of which is expected to send either beaconing data or warning messages. These procedures are classified as message-based detection mechanisms since they are the consequence of "semantic level attacks". Misbehaviour detection may also be based on the assessment of a node's trustworthiness based on the communications it transmits. These systems are referred to as node trust-based detection systems. Some current proposed methods are: False beacon information detection, False warning detection, Node trust evaluation and Feasibility assessment[9].

Misbehaviour Detection works well in conjunction with the PKI Infrastructure, as the certificate authorities can make use of the information obtained through misbehaviour detection to revoke the certificate of those agents that are damaging the network. Therefore, the Misbehaviour Authority is added to the reference architecture in Figure 10 in conjunction to the EA and AA.

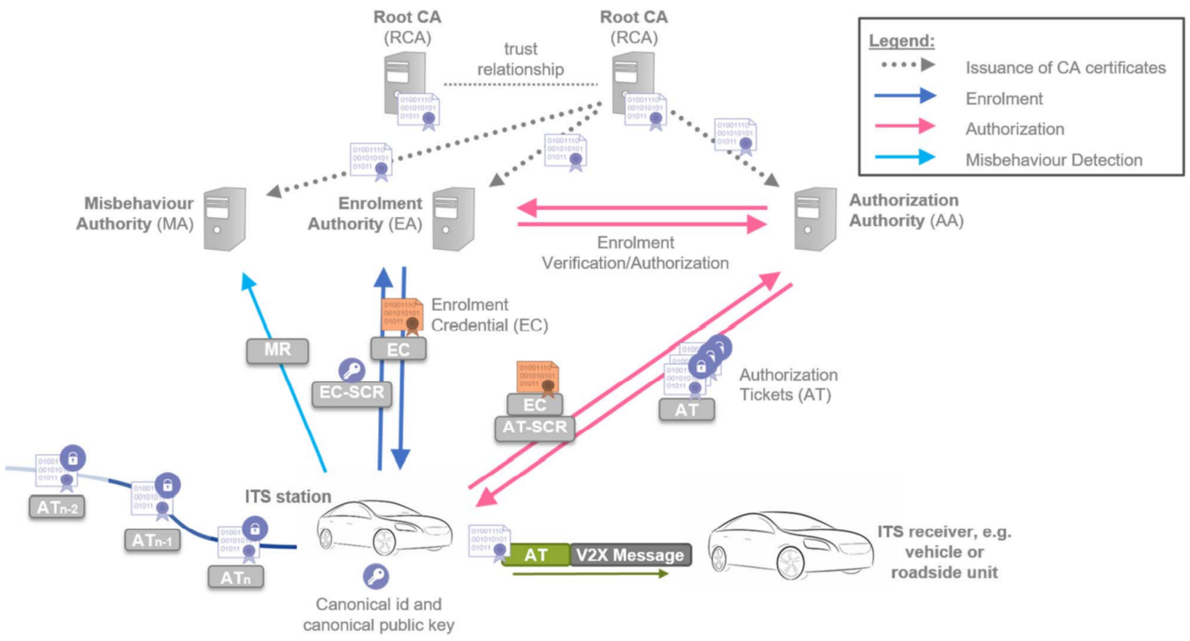

Root CA (RCA)
Root CA (RCA)
trust relationship
Legend:
Issuance of CA certificates
Enrolment
Authorization
Misbehaviour Detection
Misbehaviour Authority (MA)
Enrolment Authority (EA)
Authorization Authority (AA)
Enrolment Verification/Authorization
Enrolment Credential (EC)
MR
EC
EC-SCR
EC
AT-SCR
AT
Authorization Tickets (AT)
ATn-2
ATn-1
ATn
ITS station
Canonical id and canonical public key
AT
V2X Message
ITS receiver, e.g. vehicle or roadside unit

Figure 10: ITS Communication PKI Reference Scenario with Misbehaviour Detection [6]

3.5 Systems Engineering

Systems Engineering (SE) is a holistic and interdisciplinary approach to the design, development, and deployment of complex systems. This field has its roots in the aerospace and defense industries, where the need for sophisticated and reliable systems led to the creation of a formal methodology for systems development. Today, SE is applied to a wide range of domains, including aerospace, transportation, healthcare, information technology, and many others. This capillarity is due to the success in optimizing the performance, functionality, and safety of systems, while also addressing the needs of stakeholders and considering the broader context in which the system operates.

The literature of SE is vast and encompasses a range of topics, including requirements analysis, system architecture, system design, system integration, risk management, and system testing and evaluation. In recent years, the field has also embraced new approaches and technologies, such as Model-Based Systems Engineering or Systems-of-Systems. These last two are key aspects of this thesis motivation and implementation.

3.5.1 V-Model

The Systems Engineering process typically begins with the identification of customer requirements and the definition of system objectives. This is followed by the development of system concepts and alternatives, the selection of the preferred solution, and the detailed design of the system. Throughout the development process, Systems Engineers must manage trade-offs between technical requirements, cost, and schedule, as well as address risks and uncertainties. Once the system is developed, SE also play a critical role in its integration, testing, and commissioning[26].

This life-cycle is illustrated with the V-Model, first developed by Bröhl and Dröschel in the application area of software development in 1995. The ”V” symbol’s core meaning is the gradual breakdown of the technical system into its component parts on the left thigh and the gradual fusion of those parts into the whole system on the right thigh. The properties of the product under development are regularly tested and confirmed in addition to these two ”V” thighs. As a result, the correct system (validation) is developed in the proper way (verification)[7].

As shown in Figure 11, one of the main processes in the V-model is requirements and architecture specification. The system architecture consists of a building structure in the mechanical sense, signal flow structures and circuit diagrams in the electronic sense, as well as the structuring of a software program into its modules and components, including the respective interfaces, from a software perspective, all based on the functional structure and underlying principles of action. Hence, the system architecture embodies the components, connections, and fundamental ideas required for the creation and continued growth of a system.

The evaluation of structural alternatives is done using understandable standards, like the recognition of interdependencies, functional assessments, and taking exclusion criteria into account. This evaluation should ideally be conducted with user input. On the basis of system partitioning, an iterative process is used to establish the best feasible allocation

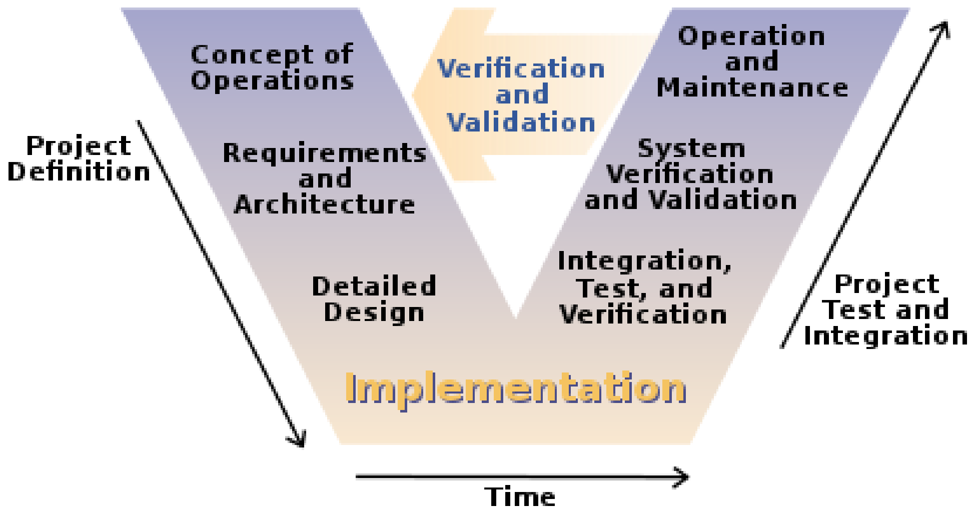

Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

Figure 11: V-model[7]

of the individual functions to the participating disciplines. The system architect's job is to break the system down into manageable components. A system is disassembled step by step from the functional, property, and implementation points of view, starting with the necessary system functions and properties. Verifiable units—system components with assigned stated needs and pertinent interface data—are the end product. The quantity and style of interfaces within an architecture and to the system boundary have a big impact on a system's simplicity, flexibility, and testability.[7]

One tool that facilitates all the mentioned tasks is the usage of a model based approach.

3.5.2 Model-Based Systems Engineering

Model-Based Systems Engineering (MBSE) is a systems engineering approach that emphasizes on the use of models to represent and manage the complexity of systems. Moreover, it is a method for designing and developing systems in a consistent, efficient, and effective manner, using graphical and mathematical representations to capture the requirements, architecture, design, and behavior of a system. In that way, MBSE provides a common understanding of a system's behavior and to facilitate communication among stakeholders.

MBSE emerged in response to the increasing complexity of modern systems, which are often composed of many interrelated components, and require the integration of multiple disciplines and technologies. The traditional approach to systems engineering, which relies primarily on text-based documentation, is often insufficient to fully capture and manage this complexity. MBSE, on the other hand, provides a visual and graphical representation of a system, enabling engineers to capture the relationships between system components and to understand the system's behavior over time.

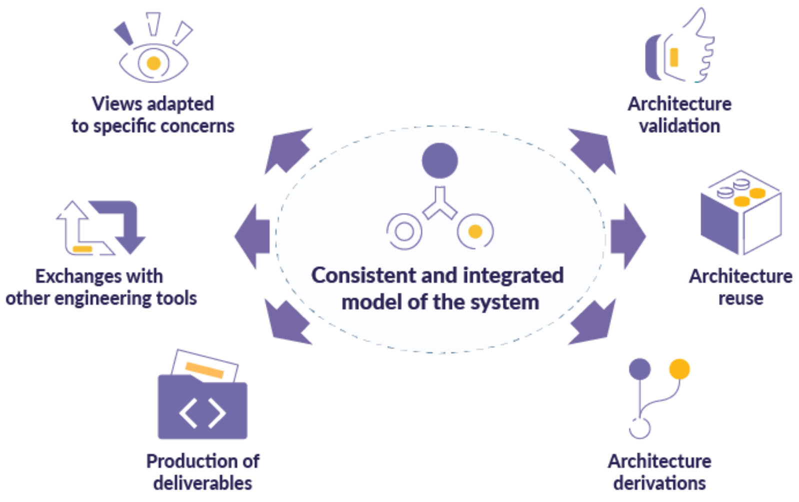

Views adapted to specific concerns
Architecture validation
Exchanges with other engineering tools
Consistent and integrated model of the system
Architecture reuse
Production of deliverables
Architecture derivations

Figure 12: Benefits of MBSE [8]

In addition to the visual representation, the use of models in MBSE provides other benefits, represented in Figure 12. First, it allows for the early detection and resolution of issues in the design process by validating the architecture, reducing the risk of costly and time-consuming changes later in the development cycle. Second, it enables engineers to reuse models among different architectures that share common functions or components. The model also opens the possibility to use other engineering tools with ease. Finally it supports collaboration and communication among stakeholders, by allowing ti have different architecture derivations that help with updates and modifications.

The MBSE process typically begins with the development of a high-level system architecture, which is represented as a set of interconnected models. These models can include functional models, which describe the behavior of the system, and physical models, which describe the physical components and their relationships. The models are then refined and developed in more detail, capturing the requirements, design, and behavior of the system. Throughout the process, the models are used to support analysis, simulation, and verification, as well as to guide the development of the system[27].

The literature of MBSE covers a range of topics, including modeling languages, modeling methodologies, model-based verification and validation, and the integration of MBSE with other systems engineering processes and practices. There is also a growing body of research exploring the use of MBSE in specific domains, such as aerospace, defense, healthcare, and transportation.

To implement the models for a MBSE Architecture, different software have been developed over the years. Among all the available options, SysML has raised as the traditional choice for companies to use.

3.5.3 Capella and Arcadia Method

The MBSE tool Capella, inspired on SysML, has been analyzed in this work and has been selected as tool of choice for this master thesis implementation. Table 2 shows a comparison between Capella and SysML illustrating their differences and similarities.

Capella is specifically designed with the goal of Define, Analyse, Design and Validate SYS/SW/HW Architectures. The intention is to be able to create a model that facilitates the systems engineering process by connecting all the people involved in the process, as it is illustrated in Figure 13. Some examples of this would be sharing proposed solutions among stakeholders, support efficient collaboration in the engineering process or allow an early evaluation and justify the Architectural Design easing the Impact Analysis.

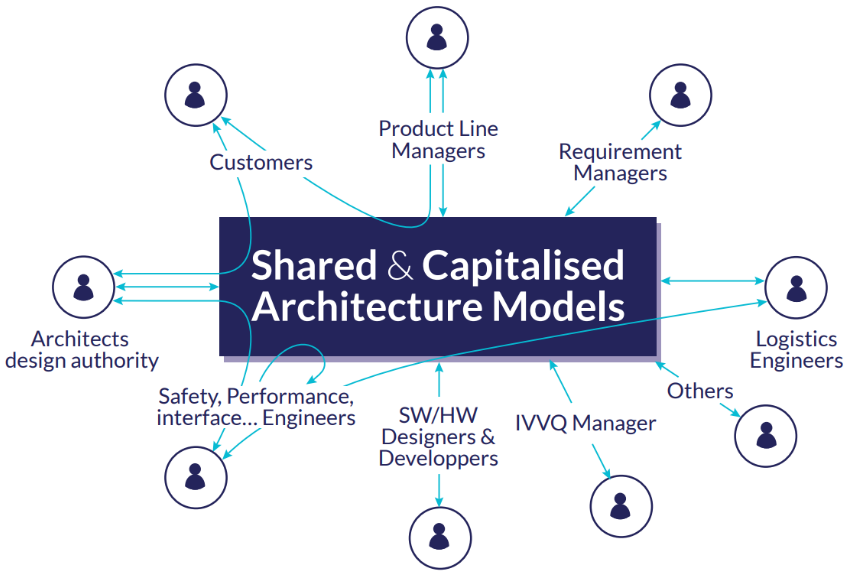

Shared & Capitalised Architecture Models
Customers
Product Line Managers
Requirement Managers
Architects design authority
Logistics Engineers
Safety, Performance, interface... Engineers
SW/HW Designers & Developpers
IVVQ Manager
Others

Figure 13: Goals of Arcadia Capella[8]

With these goals in mind, Capella proposes a development process called "Arcadia Method" where the system is developed in different levels according to different goals to fulfill. Figure 15 represents this levels into a diagram with the correspondence relationship between those.

	SysML	**Arcadia/Capella**
Positioning	A universal and uniform modeling language for modeling systems. SysML offers extremely sophisticated and rich expression tools for a wide range of applications, from high-level architecture modeling to intricate design at the cutting edge of simulation.	The Arcadia/Capella solution focuses on the creation of systems architectures, drawing inspiration from SysML concepts. The expression methods are occasionally decreased compared to SysML for the goal of a simpler learning curve and because of the precise scope addressed by Arcadia/Capella. Instead of having modeling ”experts” own the model on behalf of systems engineers, Arcadia/ultimate Capella's objective is to have systems engineers embrace the cultural shift brought about by MBSE. Therefore, the techniques and concerns of system engineering practitioners are a major driving force behind Arcadia/-Capella.
Method	Despite the fact that a variety of engineering methodologies might be used, SysML is not linked to any single technique. As a result, SysML just offers a vocabulary and makes no recommendations for how to structure models, when to apply particular concepts, etc.	The Arcadia method, in accordance with the IEEE 1220 standard and covering portions of ISO/IEC/IEEE 15288, enforces an approach structured on various engineering perspectives, establishing a clear separation between system context and need modeling (operational need analysis and system need analysis), and solution modeling (logical and physical architectures).
Language	Technically, the SysML language itself is defined as an extension of the Unified Modeling Language (UML). Both UML and SysML are general-purpose languages that target a broad range of engineering fields and rely on engineering paradigms that were developed in software utilizing techniques like types, inheritance, etc.	The UML/SysML standard and the NATO Architecture Framework (NAF) standard are most comparable to the Arcadia principles (75% and 5%, respectively). Ad-hoc imports and exports allow for interoperability with SysML tools. Some of the SysML ideas have been simplified or specialized due to the emphasis on architectural design in order to better align with the notions system engineering practitioners already use in their engineering documents and assets. For instance, this is the situation with topics connected to functional analysis.
Diagrams	SysML includes diagrams inherited from UML2 and adds new diagrams: • 4 diagrams are the same as UML2 diagrams (Sequence, State Machine, Use Case and Package); • 3 diagrams are extensions of UML2 diagrams (Activity, Block definition and Internal Block); • 2 diagrams are new diagram types (Requirement and Parametric).	Arcadia method is supported by various kinds of diagrams largely inspired by UML and SysML: • Architecture diagrams; • Data flow diagrams; • Functional chains diagrams; • Sequence diagrams; • Tree diagrams; • Mode and States diagrams; • Classes and Interfaces diagrams;

Table 2: Capella and SysML Comparison[8]

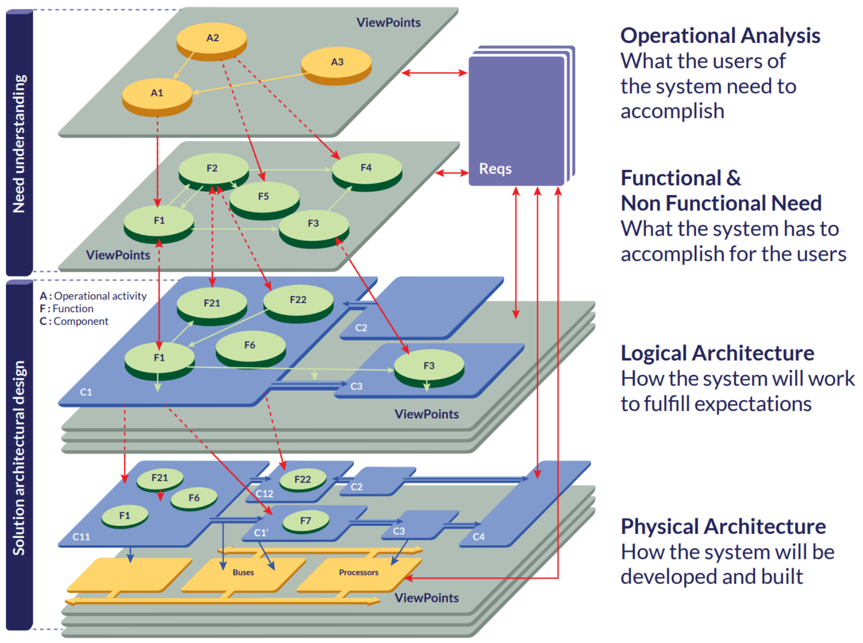

Need understanding
Solution architectural design
ViewPoints
A2
A3
A1
Reqs
F2
F4
F5
F1
F3
A : Operational activity
F : Function
C : Component
F21
F22
C2
F6
C1
C3
C12
C11
C1'
F7
C4
Buses
Processors
Operational Analysis
What the users of the system need to accomplish
Functional & Non Functional Need
What the system has to accomplish for the users
Logical Architecture
How the system will work to fulfill expectations
Physical Architecture
How the system will be developed and built

Figure 14: Capella Implementation Levels[8]

Operational Analysis In the operational analysis stage, the needs of the stakeholders and the context of the system are defined. Three main tasks are developed:

- Identify and collect stakeholder operational needs
- Specify what the system's users must accomplish
- Define entities, actors, roles, activities and concepts

The focus on this layer is to define the problem that needs to be solved for the stakeholders. In our specific case, the topic and needs were already introduced with the proposal of the thesis by Fraunhofer IKS department, and the requirements will be extracted from the ETSI standards for the implementation. Therefore, there is no need in this thesis to develop this layer in Capella as the results it offers are already present.

System Analysis In the System Analysis stage we formalize system requirements. The main tasks to perform here are the following:

- Determine the system's boundary and combine the needs and requirements
- Specify what the system needs to do for the users
- Model functional dataflows and dynamic behaviour

Notice that the word "system" is being used for the first time. It is deliberate to maintain the operational analysis step in a solution-neutral manner. In this stage the system's functions, boundaries, and interactions with the outside world start to be considered.

However, at this point the focus only remains in "What" the systems will do, leaving the "How" for the later design stages.

Logical Architecture In the Logical Architecture level the focus is on the system , its components and functions. We can perform different tasks in this level:

- Define how the system will work so as to fulfill expectations
- Specify the different components that the system is composed of
- Create State Machines for the system and components
- Model functional dataflows and dynamic behaviour
- Define data ex-changes between components

The System in the previous stage was treated as a "black box", only specifying "What" it needed to do. Now, we are prepared to define its subsystems and the functions that will then be assigned to these in order to explain "How" the needs will be achieved. During this process, necessary adjustments will be made to the system functions, such as creating more specific sub-functions, while taking into account how the various system components interact functionally.

Physical Architecture The system is viewed in its idealized form in the logical architecture. Physical components required to construct the system may not completely match logical components. During the Physical Architecture stage the system's physical architecture is created. A series of tasks are performed in this level:

- Specify how the system will be developed and built
- Software vs. hardware allocation
- Specification of interfaces, deployment configurations, trade-off analysis

In this concrete implementation an in-depth Physical Layer has not been developed as this work is intended to serve as a base for future development of a Collaborative Perception. The goal of this model is to be used by diverse projects which might use or not the same hardware configuration as others. Therefore the implementation will remain as the generic foundation upon which future development will be based on.

Each layer has different type of diagrams formed by diverse elements that help us create and show the model from multiple points of view. It is important to emphasize that the diagrams are not just drawings with boxes, but part of a model where all the elements are related to each other and can interact in multitude of ways. This is named in Capella as "Dynamic Behaviour", and it lets the user create relationships such as assigning available functions in an specific mode or create exchange diagrams using functions from a architecture diagram.

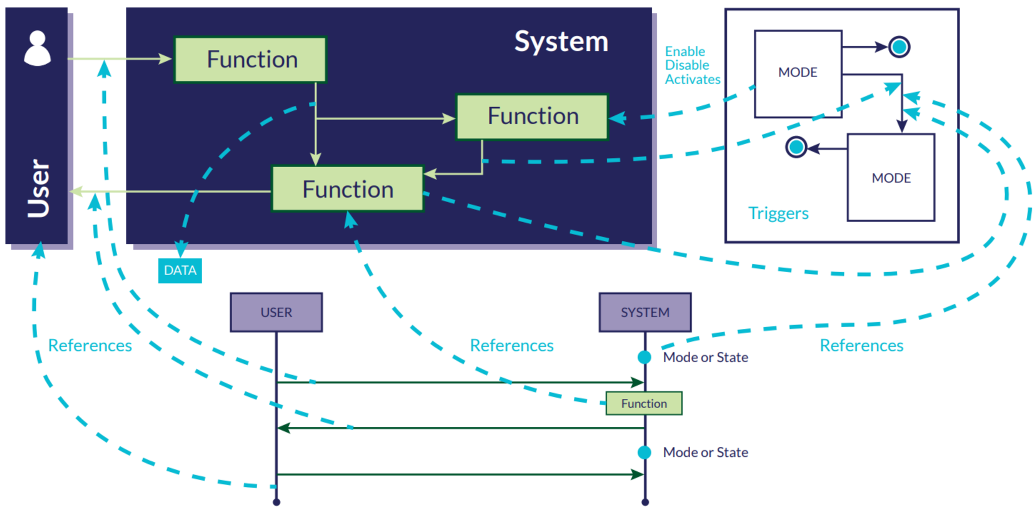


Figure 15: Capella Dynamic Behaviour[8]

4 System Design

The case study of this master thesis is a Model Based Software Architecture for Collaborative Perception in the context of Smart Cities. As introduced in section 3.3, Collaborative Perception Service (CPS) is a recent addition to the Intelligent Transport Systems (ITS) and Vehicle to Everything (V2X) communication where the aim is share the sensors acquired data and object detection between different agents to increase their performance. As the topic is still on development from the standard perspective, the creation of a model architecture is a really important step to facilitate the future implementation.

This thesis has focused on developing a scalable and modular software architecture while adhering to the aforementioned systems engineering concepts. This kind of architecture enables the reusability of already existing elements and enables an easy and quick adaption to future projects.

Our architecture will consist of a Model of the ITS Environment, like the one shown in Figure 16, consisting of various mobile ITS-S (vehicles), a fixed roadside ITS-S representing the city infrastructure and an object to be detected and whose information will be shared through the network.

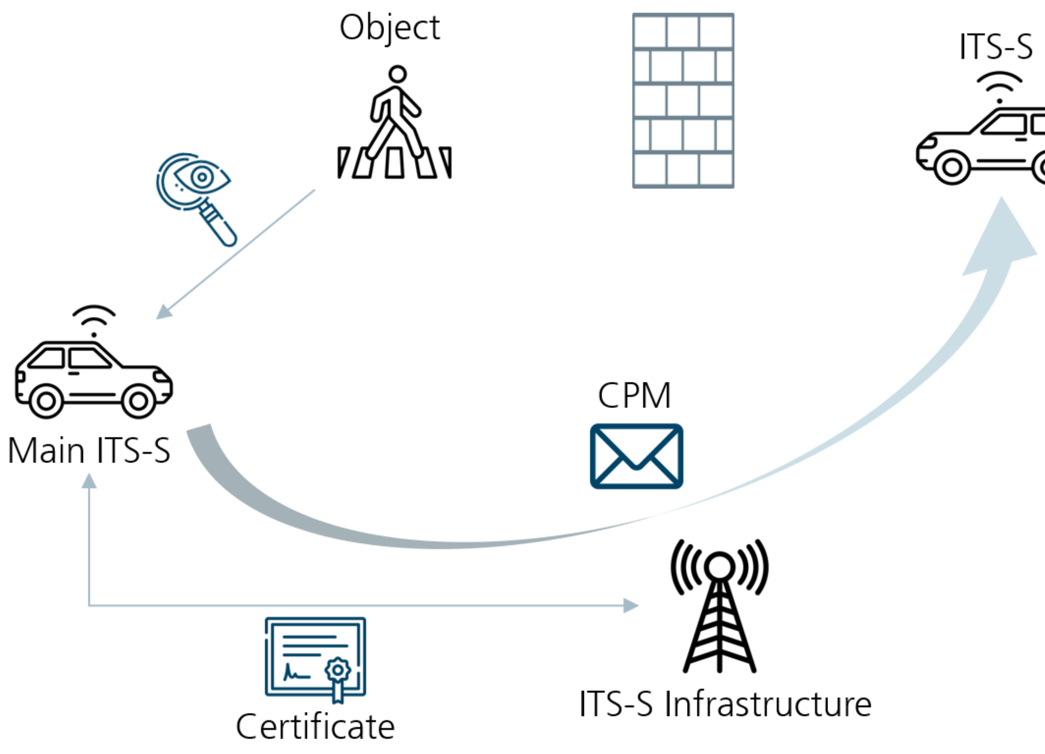


Figure 16: System Design Diagram

4.1 User Cases

A use-case is a thorough description of how a system can be utilized by a user, or a group of users, to accomplish a certain goal. To understand and explain how a system should be used and how it should respond to user needs, use-cases are frequently used in system analysis and design. The Table lists the features of the intended system:

Table 3: Use-Cases of the system.

Use-Cases	Definition	Realization
UC-1	A mobile ITS-S (vehicle) drives while processing on-board sensor data and environment data provided by an ITS Network.	**SYS-1**, **SYS-2**
UC-2	The mobile ITS-S exchanges detected objects with the other ITS-S in the Network	**SYS-3**, **SYS-4**, **SYS-5**, **SYS-6**
UC-3	The mobile ITS-S relies on a secure communication network	**SYS-7**, **SYS-8**

4.2 System Assumptions

In systems engineering, Assumptions refer to the conditions or premises that are considered to be true, without being explicitly verified or proven. These assumptions are made in order to simplify the analysis and design of a system and to facilitate decision-making. These ones can be of different types, including time, resource, quality, capacity, safety or performance constraints, among others. The assumptions of the implemented system are shown in Table 4.

Table 4: Assumptions of the system

Assumptions	Definition
AS-1	Communication Channel does not have a Bandwidth limitation
AS-2	Communication Channel does not have any package losses
AS-3	Communication Channel does not have any transmission/reception errors
AS-1	Object Detection storage limit is never reached
AS-4	Processors have always enough computing power to meet any possible timing constraint
AS-5	There is always an infrastructure in reach from mobile ITS-S
AS-6	All the infrastructure stations share the same data
AS-6	Malicious attacks can not be done directly to the infrastructure or agents

4.3 System Requirements

Functional System Requirements are the specifications that must be met for the system to work as intended and meet the needs of the users. Table 5 shows the list of requirements realizing the specified user cases.

Table 5: Functional System Requirements.

Requirements	**Definition**	**Realization**
SYS-1	System shall exchange Perception Information between Agents	**UC-1**
SYS-2	System must be able to detect objects in the enviroment	**UC-1**
SYS-3	System must store detected objects information	**UC-2**
SYS-4	System must be able to Transmit CPM	**UC-2**
SYS-5	System must be able to Receive CPM	**UC-2**
SYS-6	System shall determine its position and time	**UC-2**
SYS-7	System shall establish an Authorization based Communication	**UC-3**
SYS-8	System shall detect Misbehaviour Agents in the Network	**UC-3**

In addition, non-functional systems requirements are defined to illustrate the requirements imposed on the methodology followed for development of the system, and not the system itself. The non-functional system requirements list is shown in Table 6.

Table 6: Non-Functional System Requirements

Requirements	**Definition**
NFR-1	System is to be implemented using modelling software Capella
NFR-2	System development should follow the proposed Arcadia Method work principles
NFR-3	The components from the system must be based on existing ETSI Standards
NFR-4	System Environment shall be based on an ITS Enviroment

5 Software Architecture Development

The system architecture has been developed in two stages: System Analysis and Logical Architecture, following the Arcadia Method introduced in Section 3.5.3.

5.1 System Analysis

The system analysis stage consists on defining the context of our system and what does it need to accomplish, without getting into details of how this will be achieved.

5.1.1 System Context

The first diagram we will implement is named as System Context, where the actors that will interact with our system are defined. The main system represents one of the ITS-S, while the other possible ITS-S are represented as actor *ITS n*, leaving the possibility open for scalability. The city *Infrastructure* is constituted as another actor. Finally an *Object* actor is also defined for the detection user cases. Figure 17 shows the Capella diagram.

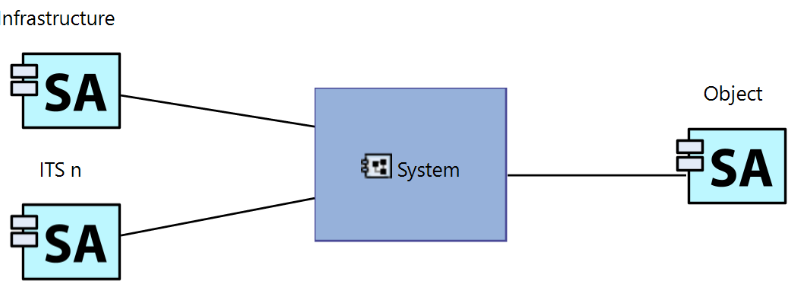


Figure 17: System Context

5.1.2 Mission and Capabilities

After having defined the actors that will interact with our system, the mission statement and capabilities need to be defined. A system's mission statement outlines the purpose, objectives, and values of the system in a short and understanding way. It provides guidance for the system's development and operation and acts as a guide for decision-making. The primary features of a mission statement are being specific, measurable, achievable, relevant, and time-bound (SMART). To guarantee its ongoing relevance, it should also be adequately communicated to all stakeholders and evaluated and updated frequently. Therefore the Mission statement is defined as: *Establish a secure Collaborative Perception Service in an ITS Environment.*

Capabilities relate to the different processes the system should fulfill to achieve the defined mission, which correlate with the main Functional System Requirements defined in 5. In Figure 18 the capabilities defined in the Capella model can be observed. There are two main capabilities that define the two major processes of our system: *Exchange Perception Information* and *Ensure Safety*. From the perception part two sub-processes are obtained: *Manage Object Detections*, which then consist of *Detect Object* and *Store Object Information*, and *Establish Communication* that separates into transmission and reception of the Collaborative Perception Messages (CPM). For the safety part two different features to ensure a secure communication are implemented: *Establish an Authorization based communication* and *Misbehaviour Detection*.

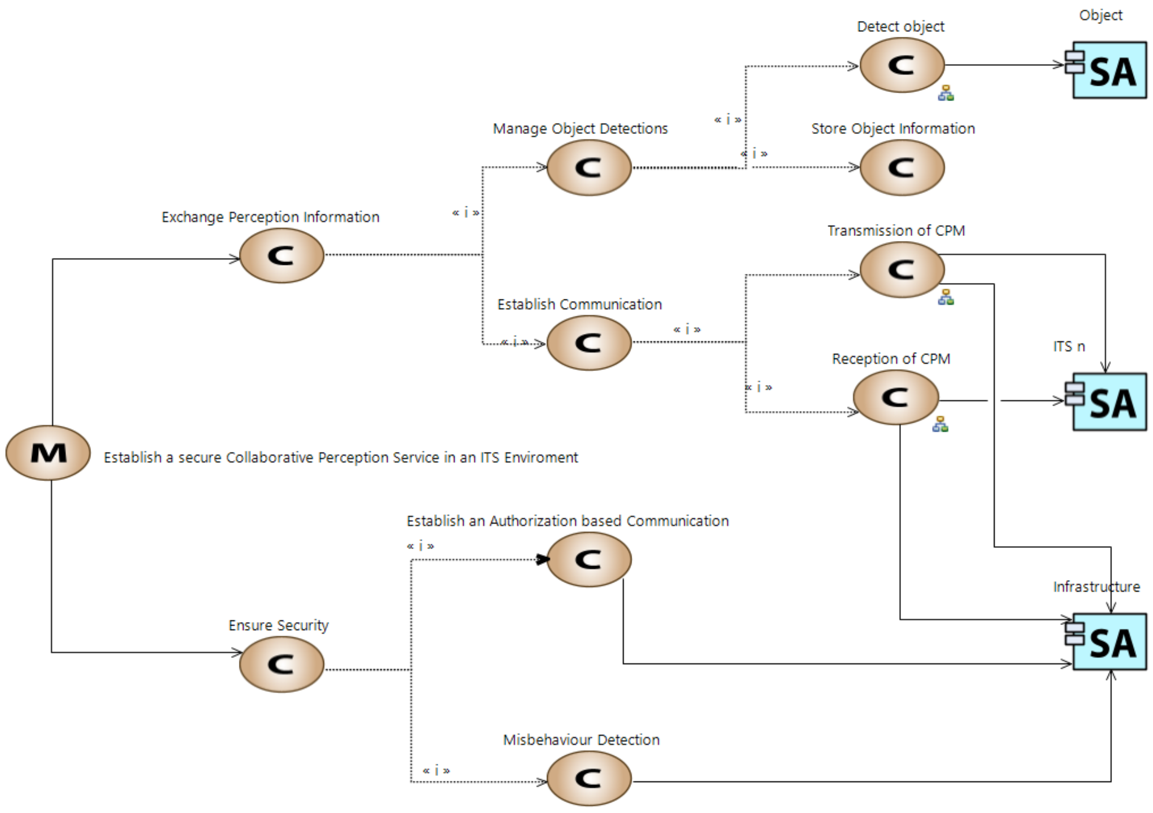


Figure 18: System Mission and Capabilities

5.1.3 Functional Breakdown

Having defined the capabilities, the next step is to add the System Requirements in the form of System Functions. Try to think abstractly while expressing the functional requirements of the system. If the results are overly detailed, the future design team's ability to come up with solutions that truly satisfy the needs is limited. It is not the goal to predict how the system will carry out its functions through system analysis, just to identifying these duties, or functional needs. For the design team, having a detailed set of functional requirements can be really beneficial.

The diagram that allocates all system functions is named as Functional Breakdown. Figure 19 shows the four Parent, or Tree, functions that have been implemented based on the main capabilities of the system. Leaf functions are then created to add further detail to those previous processes. A Leaf function is defined as the one which does not have any sub-functions. An *Enter Scene* function has also been defined for the object to be detected.

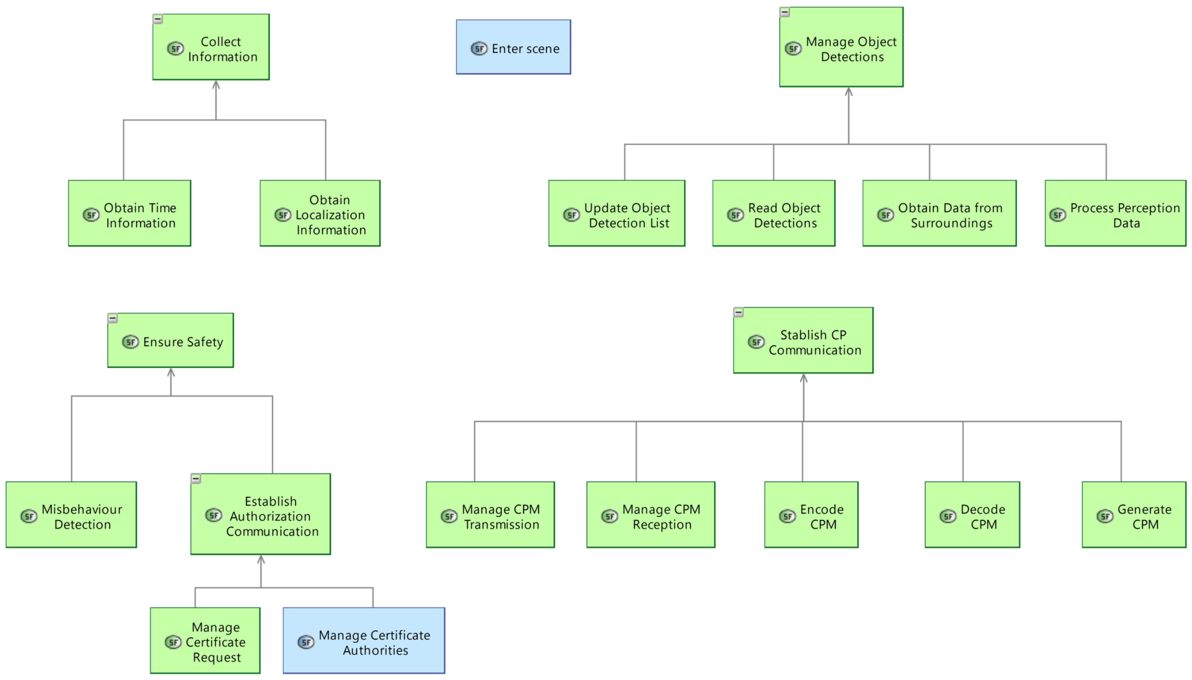


Figure 19: System Analysis Functional Breakdown

5.1.4 System Architecture

With the complete list of system functions the first Architecture Diagram can be created. These are probably the views with most information available in Capella, as it allows us to see the different functions, allocate them to different agents or components, more details about that in the Logical Architecture, and implement and observe the Functional Exchanges between them. A Functional Exchange represents the information/data that is being sent or received through the corresponding outputs and inputs of the System Functions.

Therefore, the first Architecture View is created, with the main system representing a mobile agent, and the three external actors, the infrastructure, ITS n and Object. Next step is to allocate the functions to the elements, allowing then to show them in the diagram, as it is done in Figure 20 with the tree functions.

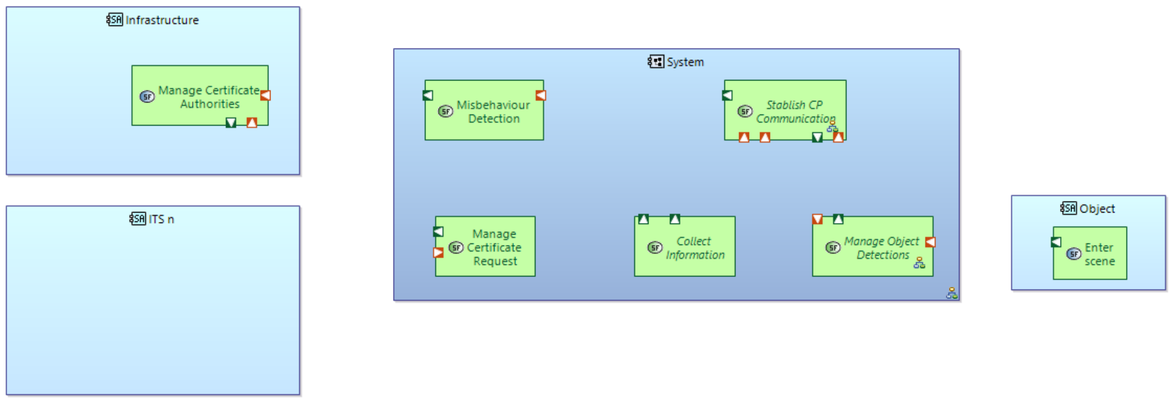


Figure 20: System Architecture Tree Functions

However, in Capella, functional exchanges must be defined only at the lowest possible leaf functions. Therefore, a view with these sub-functions is created.

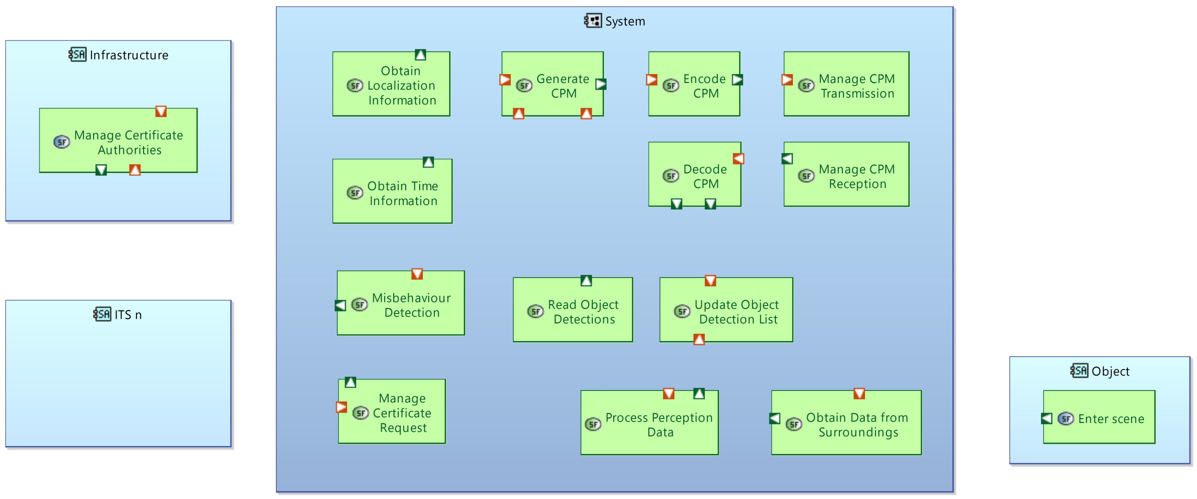


Figure 21: System Architecture Leaf Functions

With the leaf functions in place, the Functional Exchanges between them can be defined. Functional Exchanges are defined as the information/data that is shared between two functions. The collection of Functional Exchanges from the System Analysis level can be found on the following Table 7.

Table 7: System Analysis Functional Exchanges

Functional Exchange	**Source Function**	**Target Function**
Physical presence	Enter Scene	Obtain data from surroundings
Localization data	Obtain Localization Information	Generate CPM
Time data	Obtain Localization Information	Generate CPM
Object information	Read Object Detectons	Generate CPM
Received CPM	Manage CPM Reception	Decode CPM
Received object information	Decode CPM	Misbehaviour Detection
Received object information	Decode CPM	Update Object Detection List
Complete CPM	Generate CPM	Encode CPM
Encoded CPM	Encode CM	Manage CPM Transmission
Certificate request	Manage Certificate Request	Manage Certificate Authorities
Valid certificate	Manage Certificate Authorities	Manage Certificate Request
Misbehaviour Report	Misbehaviour Detection	Manage Certificate Authorities

Figure 22 shows the resulting architecture view after the addition of the Functional Exchanges.

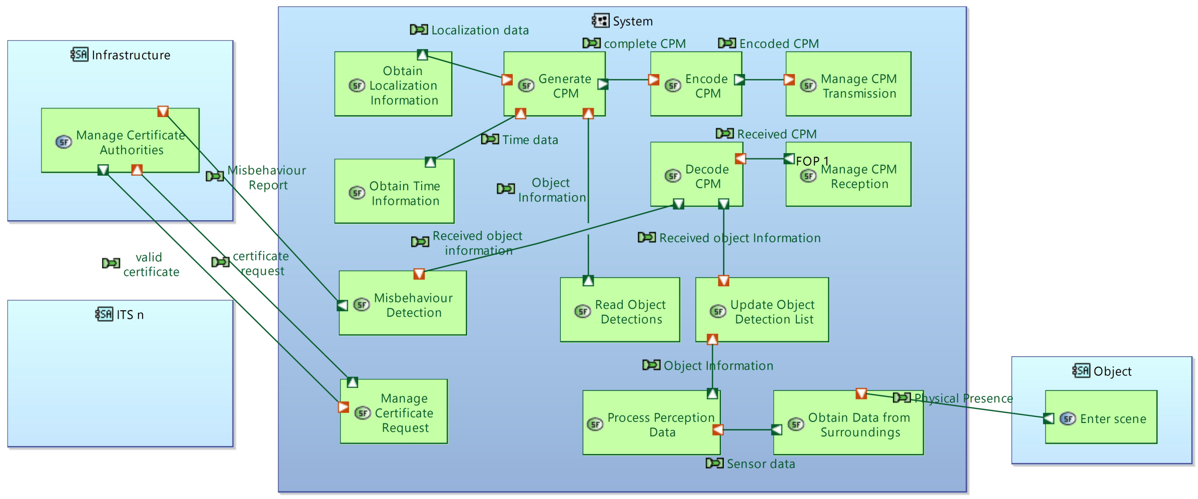


Figure 22: System Architecture Leaf Functions with Functional Exchanges

As mentioned before, the previous steps are not just building diagrams, but a model. Therefore, if the previous diagram where there were only the Parent functions are checked, it is observed how the Functional exchanges also appear, but not all of them. Only the functional exchanges that happen between functions under different parents show, and the "internal" exchanges that take place inside a main function remain hidden, as seen in the following Figure 23.

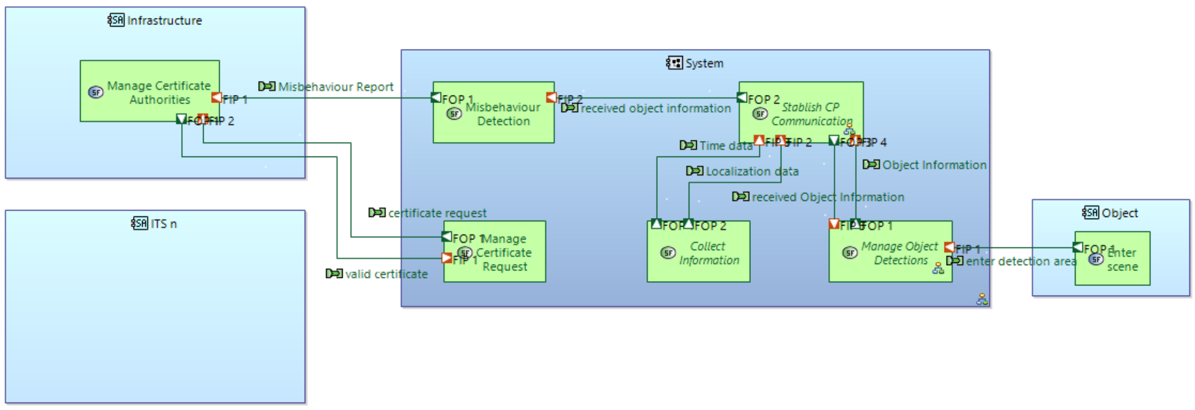


Figure 23: System Architecture Tree Functions with Functional Exchanges

The remaining step is to create a duplicate of the system to allocate the functions in the *ITSn* actor and *Infrastructure*, giving them the capabilities of an ITS-S. The main system works as the main development point from which the model rest of ITS-S will be copied. This is know in MBSE as Replication, and Capella offers specific tools to perform the process. The user is able to create a REC (Replicable Elements Collection) from any collection of elements in Capella which can be used to create as many Replica (RPL) as desired. In addition, RPL Instantiating is not limited to the current project, it is possible to make usage of REC from other projects, opening the possibility to create Libraries to increase the ease of reusability of developed Models. Instantiated RPLs keep the reference from the original REC, which allows to update them if changes are done to the REC.

Therefore, a REC containing the functions and exchanges from the main System is created and replicated as an RPL allocated in actors *ITSn* and *Infrastructure*. At the time of creating the RPL, a Suffix can be added to differentiate the duplicated functions from the originals. The results of this operation are shown in the Results Section 6.1.

5.2 Logical Architecture

Having completed the System Analysis stage it has been defined what are the functional requirements of "What" the system should do. Now, in the Logical Architecture level, the focus will be on detailing this requirements to specify "How" these requirements will be meet. To do that, the main tool will be to create more specific and technical sub-functions.

Also important to notice, all the newly added functional requirements from this section are based on an specific ETSI Standard. This will be really useful for future implementation stages, where the design team will have he standards as a clear reference and it will ensure that the model can be adopted not only by a specific project, but as a widely used starting point for any one working with those standards.

5.2.1 Components

In this stage, behavioral study is carried out and the subsystems are established. A subsystem (referred to as a component in Capella) is a group of a system's components that performs a distinct function on its own. Component and interface definition assists in the partition of design activities into smaller jobs that may be given to various development teams.

The components of this architecture are based on the ITS Architecture, present in most of ITS related standards, such as ETSI EN 302 665[5] or ETSI TS 103 324[6], shown in Figure 6 and explained in Section 3.4.

Each of the layers of the ITS Station will be converted into a Component. In addition, the facilities layer will also be divided into different sub-components depending on their nature. To select these sub-components a literature research has been carried out into different standards in order to find a good fit that could allocate the functions from systems analysis and expand them into a more detailed sub-functions. The result of this research is summarized in Table 8 while on Figure 24 we can observe the Component Breakdown tree diagram showing all the parent and leaf components.

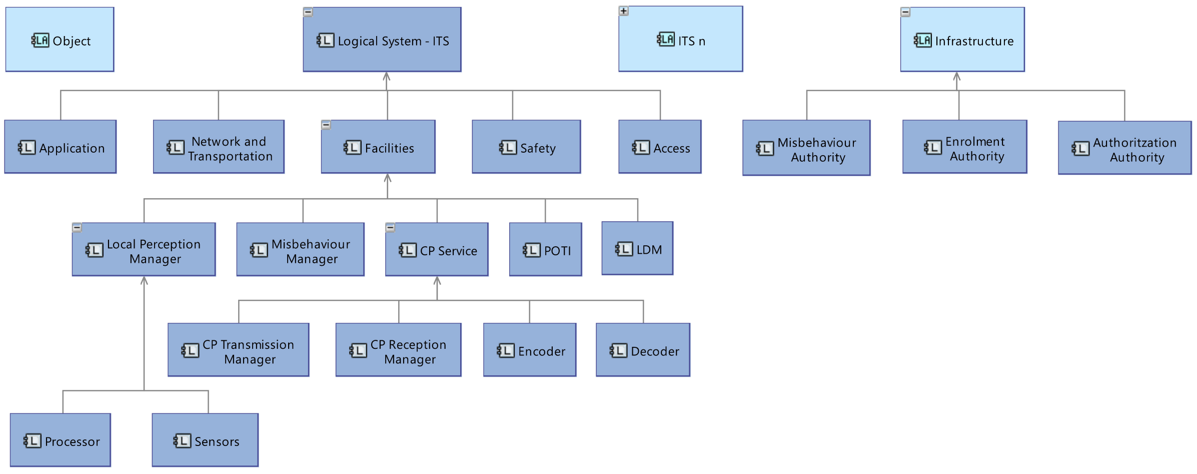


Figure 24: Logical Architecture Component Breakdown

Table 8: Logical Architecture Component List and Related Standard

Component	**Parent**	**Used Standard**
ITS Station	-	ETSI EN 302 665 [5]
Application	ITS Station	ETSI TS 101 53 [28]
Network and Transportation	ITS Station	ETSI EN 302 636-5 [23] & ETSI TS 103 248 [24]
Access	ITS Station	ETSI EN 302 665 [5]
Safety	ITS Station	ETSI TS 102 940 [25]
Facilities	ITS Station	ETSI TS 102 894-1 [29]
Local Perception Manager	Facilities	-
Processor	Local Perception Manager	-
Sensors	Local Perception Manager	-
Misbehaviour Manager	Facilities	ETSI TS 103 460 [9] & ETSI TS 103 759 [30]
CP Service	Facilities	ETSI TS 103 324 [6]
CP Tranmission Manager	CP Service	ETSI TS 103 324 [6]
CP Reception Manager	CP Service	ETSI TS 103 324 [6]
Encoder	CP Service	ETSI TS 103 324 [6]
Decoder	CP Service	ETSI TS 103 324 [6]
POTI	Facilities	ETSI EN 302 890 [22]
LDM	Facilities	ETSI EN 302 895 [31]
Infrastructure	-	ETSI EN 102 940 [25]
Enrolment Authority	Infrastructure	ETSI EN 102 940 [25]
Authorization Authority	Infrastructure	ETSI EN 102 940 [25]
Misbehaviour Authority	Infrastructure	ETSI EN 103 460 [9]

Local Perception related components do not have an assigned standard, as the expansion on this topic is out of the scope of this thesis.

With the defined components, the first Logical Architecture is created. It is similar to the one from system analysis stage, but now the system and actors have allocated inside them the components and sub-components, as seen in Figure 25.

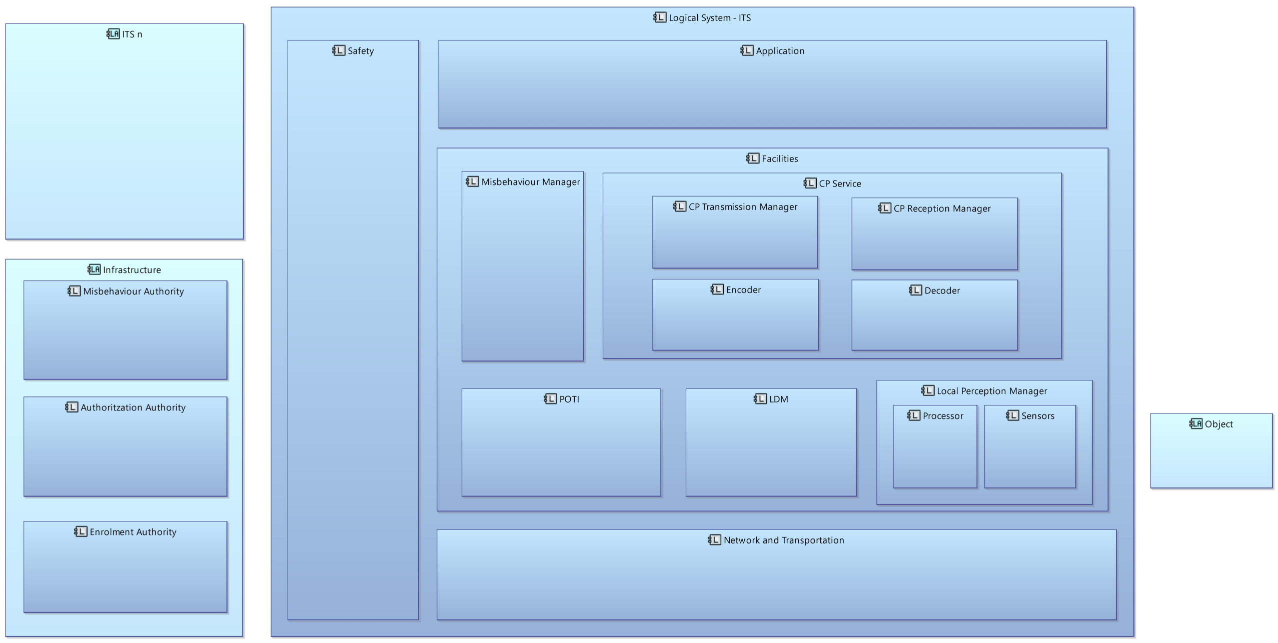


Figure 25: Logical Architecture Components Only

In the following subsections each component is explained in detail to show which processes have been implemented to illustrate how our system will operate.

5.2.2 Manage Object Detection

As mentioned, the Local Perception sensors and image processing are out of the scope of this thesis. Therefore, the functions remain the same from the previous stage and they are only allocated to their specific subcomponents. This is done to enable SoS Engineering for future developments, where a whole new project could be created for the sensor fusion processing part.

To address the storage of object detection information two functions were created at the System Analysis stage: *Update Object Detection List* and *Read Object Detection*. These two functions illustrate well what is the goal, in this case, to have a data base in form of a list of objects in the environment over time and to be able to update or read a value from this list.

A concrete solution to this goal can be achieved by the usage of a *Local Dynamic Map*, specified in ETSI 302 895 [31]. This component is placed at the facilities layer and divides the main tasks in two functions: *LDM Service* and *LDM Maintenance*.

The *LDM Service* is connected to authorized *LDM Data Providers*, who provide information to the LDM, and *LDM Data Consumers*, to whom the data is made available by the LDM. The LDM has three different type of interface ports:

- Transaction interface where the data of new detections is being exchanged with the Data Providers

- Two type of interfaces for Data Consumers
  - Query interface for single time request
  - Publish/Subscribe interface for Data Consumers that require a periodic or event based service of data.

CPM Messages are generated periodically, so the interface used to send data to the *CP Service* facilities will be a subscribe type.

Apart from that, LDM shall allow applications and other facilities to register and deregister as *LDM Data Providers* or *LDM Data Consumers*. It must also verify the authorization of *Data Providers or Consumers* prior to any data access. This authorization can be revoked at any moment by the security service.

LDM Maintanance is in charge of the long term storage of the Object Data. Data Object must be mantained through their time validity and withitn the LDM Area of Manitenance.

The LDM considers a *LDM Data Object* to be valid during the time period starting on the timestamp of the *LDM Data Object* and for the duration of the time validity period. A *LDM Data Provider* shall specify the timestamp value, specified when adding or updating the *Data Object*, and the time validity, which has a default value set on registration that can be overwritten if it is specified at the addition or update of the Data Object, of every *LDM Data Object* it provides to the LDM.

If an *LDM Data Object*'s position crosses over with the *LDM Area of Maintenance*, the LDM will consider it to be valid. Whenever an *LDM Data Object* is added or updated, the corresponding *LDM Data Provider* provides the location. The *LDM Area of Maintenance* is a territory that can be specified in relation to the current position of the host ITS-S.

Therefore, these new two functions replace the ones from the system analysis level. In Figure 26 the resulting architecture can be observed, with the Actor, Local Perception Manager and LDM components.

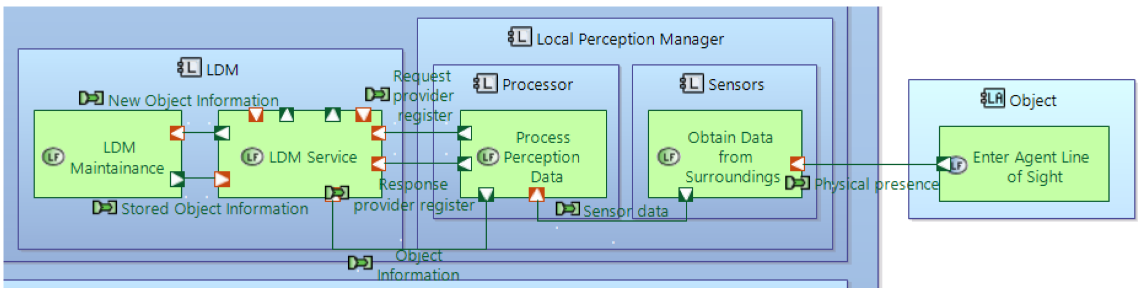


Figure 26: Manage Object Detection Architecture

The same functional exchange to request registration/subscription serves to register, deregister or revoke the permission. The complete list of functions and exchanges can be found in Table 9.

Table 9: Object Detection Functional Exchanges

Functional Exchange	Source Function	Target Function
Physical presence	Enter Scene	Obtain data from surroundings
Sensor Data	Obtain Data from Surroundings	Process Perception Data
Request provider register	Process Perception Data	LDM Service
Response provider register	LDM Service Process	Perception Data
Object information	Process Perception Data	LDM Service
New Object information	LDM Service	LDM Maintenance
Stored Object information	LDM Maintenance	LDM Service

5.2.3 Position and Time

Time and Position Management remain the same functions from the System Analysis stage. The information is requested as a query and return to the consumer. To expand on this topic, specific information can be found in ETSI 302 890 [22].

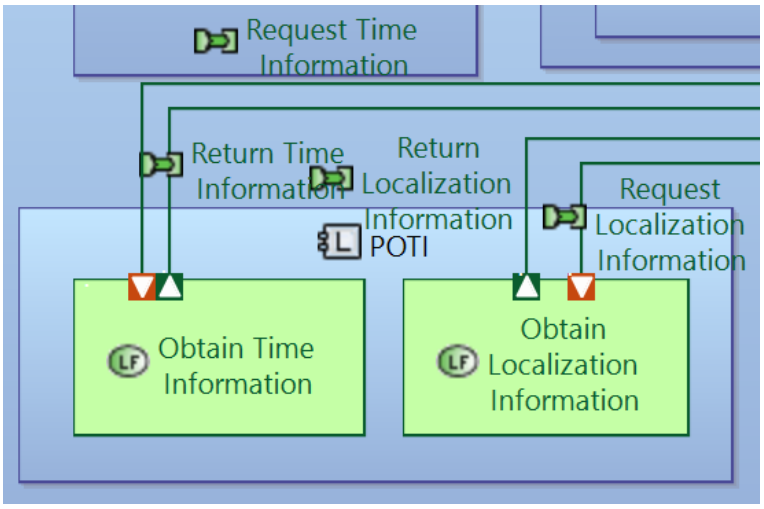

Request Time Information
Return Time Information
Return Localization Information
Request Localization Information
POTI
Obtain Time Information
Obtain Localization Information

Figure 27: Position and Time Management

5.2.4 CP Service - Transmission

The process of generating Collective Perception Messages (CPMs), as defined in ETSI TS 103 324 [6], involves the creation and transmission of CPMs by the originating Intelligent Transport System Station (ITS-S), which are then disseminated through the ITS networking and transport layer using the communication system in use. CPMs are generated periodically by the ITS-S, with the rate controlled by the CP service. During each generation event, the ITS-S takes into account various factors, such as the dynamic behavior of detected objects, the reception of CPMs from other ITS-Ss, potential receiver's ability to compute detected free spaces based on the transmitted data, and the radio channel load to create a new CPM.

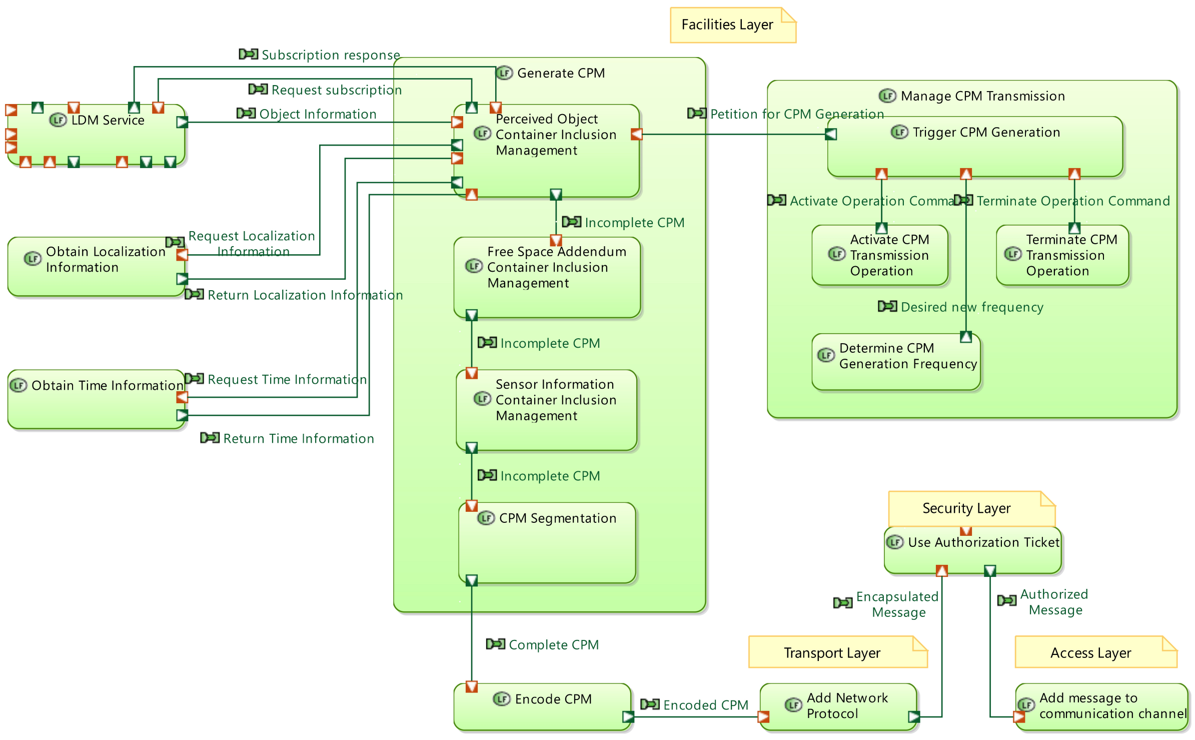
Therefore, the standard matches with the division of processes from the systems analysis, where the transmission process was divided into three main functions: *Manage CPM Transmission*, *Generate CPM* and *Encode CPM*. The Logical Architecture stage adds new subfunctions to expand on these three processes, in addition to newly added functionalities to complement, all taken from the standard specification and requirements.

The *Manage CPM Transmission* oversees the following subtasks: *Activate or terminate the CPM Transmission operation*, *Determine the frequency of CPM Generation* and *Trigger the CPM Generation*.

On the other hand, the *CPM Generation* consists of all the various steps needed to construct a *CPM Message Containers*, which have been presented in Section 3.4.1. As a result the *CPM Generation* is divided in the following subfunctions: *Perceived Object Container Inclusion Management*, *Free Space Addendum Container Inclusion Management*, *Sensor Information Container Inclusion Management*, *CPM Segmentation*.

The functional chain will consist then of first the *Transmission Manager* activating the operation, setting up the frequency, and requesting the periodic generation. Then, the CPM is created by adding one by one the containers. Before leaving the *Facilities Layer* the message is first encoded through the *Encode CPM* function to then travel through the *Network Layer*, where the protocol is added. The used *BTP Protocol* is explained in Section 3.4.4. Before the message is sent through the communication channel at the *Access Layer*, the *Authorization Ticket* must be added to the message. Therefore the encoded message goes first to *Use Authorization Ticket* and then to *Add message to communication channel*, where the transmission process finished. Details on how the AT are obtained are explained in Section 5.2.6.

Figure 28 illustrates the Logical Data Flow diagram for *CPM Transmission and Generation*. This type of diagram allows the user to showcase the different exchanges between different functions.



Facilities Layer
Subscription response
Request subscription
Object Information
Generate CPM
LDM Service
Perceived Object Container Inclusion Management
Petition for CPM Generation
Manage CPM Transmission
Trigger CPM Generation
Activate Operation Command
Terminate Operation Command
Activate CPM Transmission Operation
Terminate CPM Transmission Operation
Desired new frequency
Determine CPM Generation Frequency
Incomplete CPM
Obtain Localization Information
Request Localization Information
Return Localization Information
Free Space Addendum Container Inclusion Management
Incomplete CPM
Obtain Time Information
Request Time Information
Return Time Information
Sensor Information Container Inclusion Management
Incomplete CPM
CPM Segmentation
Security Layer
Use Authorization Ticket
Encapsulated Message
Authorized Message
Transport Layer
Access Layer
Complete CPM
Encode CPM
Encoded CPM
Add Network Protocol
Add message to communication channel

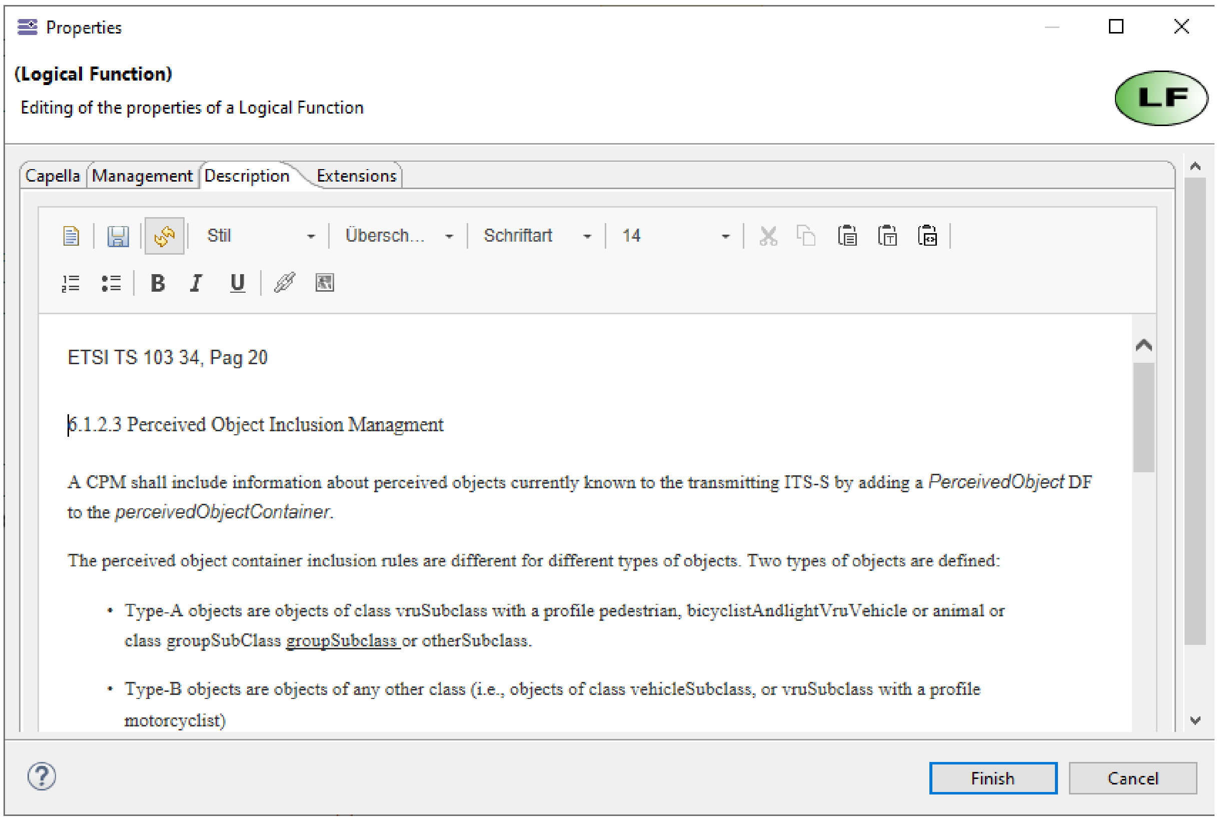
Figure 28: CPM Transmission Logical Data Flow

The following Table 10 contains all the functional exchanges that take place in the CPM Transmission Logical Data Flow from Figure 28.

Table 10: CPM Transmission Functional Exchanges

Functional Exchange	**Source Function**	**Target Function**
Desired new frequency	Determine CPM Generation Frequency	Trigger CPM Generation
Activate Operation Command	Activate CPM Transmission Operation	Trigger CPM Generation
Terminate Operation Command	Terminate CPM Transmission Operation	Trigger CPM Generation
Petition for CPM Generation	Trigger CPM Generation	Perceived Object Container Inclusion Management
Subscription request	Perceived Object Container Inclusion Management	LDM Service
Subscription response	LDM Service	Perceived Object Container Inclusion Management
Object information	LDM Service	Perceived Object Container Inclusion Management
Request Localization Information	Perceived Object Container Inclusion Management	Obtain Localization Information
Return Localization Information	Obtain Localization Information	Perceived Object Container Inclusion Management
Request Time Information	Perceived Object Container Inclusion Management	Obtain Time Information
Return Time Information	Obtain Time Information	Perceived Object Container Inclusion Management
Incomplete CPM	Perceived Object Container Information Management	Free Space Addendrum Container Inclusion Management
Incomplete CPM	Free Space Addendrum Container Inclusion Management	Sensor Information Container Inclusion Management
Incomplete CPM	Sensor Information Container Inclusion Management	CPM Segmentation
Complete CPM	CPM Segmentation	Encode CPM
Encoded CPM	Encode CPM	Add Network Protocol
Encapsulated Message	Add Network Protocol	Use Authorization Ticket
Authorized Message	Use Authorization Ticket	Add message to communication channel

In the cases where the modelled function is of high complexity and has technical requirements that should be taken into consideration when implementing such function, these requirements are added into the description of the function, so that the future design team have them as a direct reference at all time. For example, in Figure 29 the description of the function *Perceived Object Inclusion Management* is shown, where the information from the standard, introduced in Section 3.4.1, is added. This procedure had been done with almost all the functions in the logical architecture.



ETSI TS 103 34, Pag 20

6.1.2.3 Perceived Object Inclusion Managment

A CPM shall include information about perceived objects currently known to the transmitting ITS-S by adding a *PerceivedObject* DF to the *perceivedObjectContainer*.

The perceived object container inclusion rules are different for different types of objects. Two types of objects are defined:

- Type-A objects are objects of class vruSubclass with a profile pedestrian, bicyclistAndlightVruVehicle or animal or class groupSubClass groupSubclass or otherSubclass.
- Type-B objects are objects of any other class (i.e., objects of class vehicleSubclass, or vruSubclass with a profile motorcyclist)

Figure 29: Perceived Object Inclusion Management Description in Capella

5.2.5 CP Service - Reception

The function Manage CPM Reception is in charge of two tasks defined in ETSI TS 103 324 [6]: Triggering the decoding of the CPM upon reception of a CPM and Provisioning of the decoded CPM to the LDM and other applications. Therefore, two new sub-functions: *Provide CPM* and *Trigger CPM Decoding* are created for the Reception Manager.

The process of receiving a message is the opposite as the one described for the transmission. The message is first received at the *Access Layer* through the function Receive message from communication channel. Then, this received message is sent to the *Security Layer*, where its PKI credentials, both the Enrolment Certificate and adequate Authorization Ticket are verified. If correct, the verified message goes through the *Transportation Layer*, where function *Substract Network Protocol* identifies the type of message and shares it with the corresponding facility.

When the message arrives to the *CP Service*, the *CPM Reception Manager* triggers its decoding. The message is then sent to *Decode CPM* and is returned after the process is finished. Finally, the CPM is provided to the other facilities that require it, working as a

Data Provider for the LDM and also facilitating the message to the *Local Misbehaviour Detection*. The process of data provisioning and registration as *Data Provider* is already explained in Section 5.2.2. It is also worth mentioning that *Local Misbehaviour* will also have the *LDM Service* as data provider for the inputs apart from the *Provide CPM*. The *Misbehaviour Detection* is developed in detail in Section 5.2.7. Figure 30 shows the final Logical Data Flow diagram with all the functions that take place in the reception of CPM.

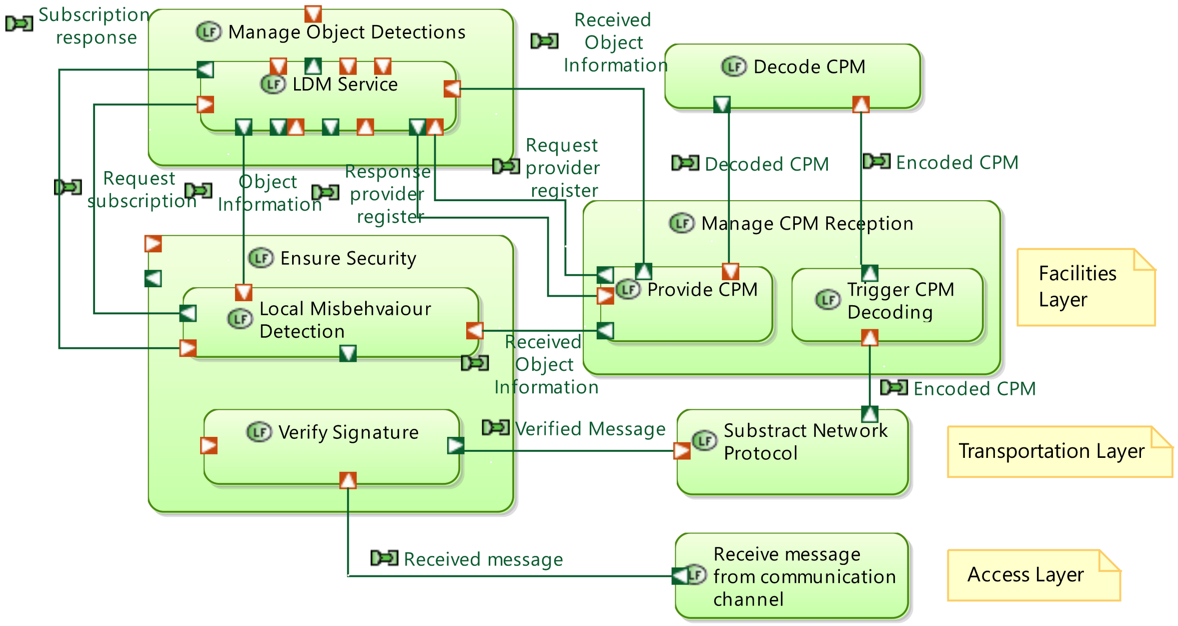


Figure 30: CPM Reception Logical Data Flow

Following Table 11 contains all the functional exchanges from Figure 30.

Table 11: CPM Reception Functional Exchanges

Functional Exchange	**Source Function**	**Target Function**
Received message	Receive message from communication channel	Verify Signature
Verified Message	Verify Signature	Substract Network Protocol
Encoded CPM	Substract Network Protocol	Trigger CPM Decoding
Encoded CPM	Trigger CPM Decoding	Decode CPM
Decoded CPM	Decode CPM	Provide CPM
Request provider register	Provide CPM	LDM Service
Response provider register	LDM Service	Provide CPM
Received Object Information	Provide CPM	LDM Service
Received Object Information	Provide CPM	Local Misbehaviour Detection
Object Information	LDM Service	Local Misbehaviour Detection
Request subscription	Local Misbehaviour Detection	LDM Service
Subscription response	LDM Service	Local Misbehaviour Detection

5.2.6 Authorization Based Communication

The need to ensure a secure communication in the ITS network can be pursued by different methods. The selected solution for this implementation has been a Public Key Infrastructure. Diverse subfunctions are created to manage the certificate both at the individual ITS-S, under the parent function *Manage Certificate Request*, and at the infrastructure, from the parent function *Manage Certificate Authorities*, where the different Authorities are located. These subfunctions take part in the different steps that need to be followed to be granted access to the network [25].

Initialization First step to be followed is the *Initialization.* This function makes use information present in the vehicle such as the canonical identity, a unique and immutable value established by the manufacturer, in addition to information provided by the network through the *Provisioning* function, to create a set of credentials know as *Canonical Credentials.*

Enrolment The *Enrolment* function uses the *Canonical Credentials* to request the *Enrolment Credentials* (EC) to the *Enrolment Authority* (EA). If the request is accepted, the EA will return the correspondent EC to the agent. This process can be invoked again when the end of life time of the EC is reached or if it is revoked for any other case.

Authorization Having a valid EC, grants the possibility to the *Authorization* function to request *Authorization Tickets* (AT). Before accepting the request, the *Authorization Authority* (AA) might request the EA to perform a validation on the received EC, to ensure it corresponds to a valid and certified agent. If the EA and AA accept the request, the ITS-S will receive AT to be used in an specific application or functionality. These tickets are single-use and are expelled through the *Use Authorization Ticket* function, which is part of the CPM transmission cycle described in Section 5.2.4.

From the EA perspective, the certificates can be revoked at any given time. One of the possible causes is the inclusion of a given ITS-S into the *Certificate Revocation List* issued by the *Misbehaviour Authority* (MA). This lists includes all the agents that haven been determined to have acted with misbehaviour and therefore are banned from the network. How *Misbehaviour Detection* and *Misbehaviour Report* work is explained in the following Section 5.2.7.

The following Figure 31 shows the Logical Data Flow diagram containing all the functions involved in the initialization, issuing, usage and revocation of certificates.

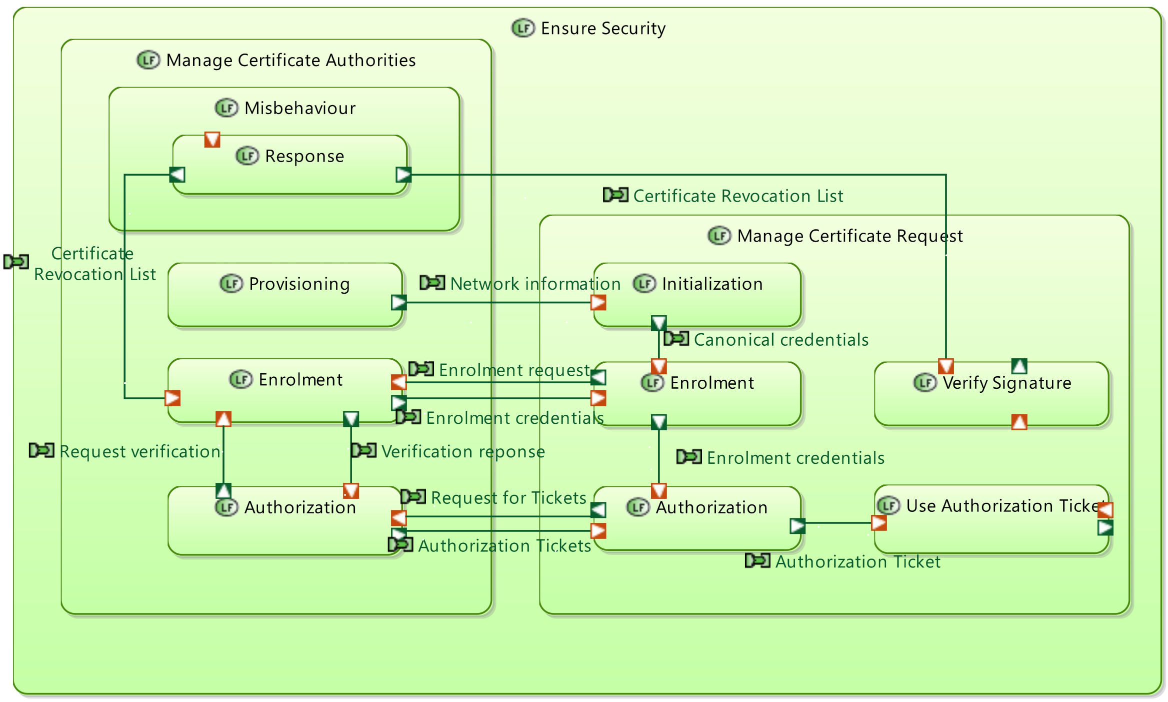


Figure 31: Certificate Issuing Logical Data Flow

Table 12 contains the functional exchanges from Figure 31.

Table 12: Certificate Issuing Functional Exchanges

Functional Exchange	**Source Function**	**Target Function**
Network information	Provisioning	Initialization (ITS-S)
Canonical Credentials	Initialization (ITS-S)	Enrolment (ITS-S)
Enrolment request	Enrolment (ITS-S)	Enrolment (Infrastructure)
Enrolment credentials	Enrolment (Infrastructure)	Enrolment (ITS-S)
Enrolment credentials	Enrolment (ITS-S)	Authorization (ITS-S)
Request for Tickets	Authorization (ITS-S)	Authorization (Infrastructure)
Authorization Tickets	Authorization (Infrastructure)	Authorization (ITS-S)
Request verification	Authorization (Infrastructure)	Enrolment (Infrastructure)
Verification response	Enrolment (Infrastructure)	Authorization (Infrastructure)
Authorization Ticket	Authorization (ITS-S)	Use Authorization Ticket
Certificate Revocation List	Response	Verify Signature
Certificate Revocation List	Response	Enrolment (Infrastructure)

5.2.7 Misbehaviour Detection

Misbehaviour Detection works closely with the other facilities to add another layer of safety to the authorization based communication. The goal is to compare the received data with the local detections to check for errors that could lead to a potential malfunctioning or malware agent in the network. However, the vehicle ITS-S does not have the granted power to revoke certificates itself, therefore it will need to report those detected cases to the *Misbehaviour Agent* (MA) which is part of the *Infrastructure* that will take further actions if needed.

To achieve this goal, different subfunctions are created to model the requirements and procedures specified in ETSI 103 460 [9]. The complete cycle can be divided into three main steps: *Local Misbehaviour Detection*, *Misbehaviour Reporting* and *Misbehaviour Agent*.

Local Misbehaviour Detection First, the *Local Misbehaviour Detection* receives the input data from the LDM and *CPM Reception Manager*, through the functions *Provide CPM* and *LDM Service* explained in their respective sections. The subfunction *Detectors* is in charge of comparing the input data to find inconsistencies between the received messages and the local detections. Diverse detection strategies are proposed, although there is still not a defined method for the CP Standard. However, there is a potential misbehaviour detection mechanism list for CAMs, which has been added in the Appendice C. When an inconsistency is detected, the information is send to the *Event Categorization* where it is classified into different type of errors or attacks. The category alongside the previous information is then send to the *Misbehaviour Reporting* function.

Misbehaviour Reporting Diverse functions are involved in the creation of the *Misbehaviour Report*(MR):

- Decision: concludes the necessity to generate or not a MR for the event observed in the Local Misbehaviour Detection
- Generation: uses the data provided by the local misbehavior detection service and, optionally, additional information collected from the State to construct the MR. It also chooses whether to send the report to *Storage* or *Transmission.*
- Transmission: in charge sending the reports to the *Misbehaviour Authority.* If there are multiple reports ready to be shared, *Transmission* selects which ones to send and the order.
- Storage: saves MR provided by *Generation* and provides them to *Transmission* when requested, deleting old reports.
- State: holds data needed by the other functions to perform their tasks and manages three separate budgets: one for the report creation (processing time limits), the second one for the storage of reports (total volume of generated reports might exceed the available storage) and a third for transmission (as there might not be always an available connection).

Misbehaviour Authority All the MR sent to the MA are stored in a database in function *Misbehaviour Collection.* Then, *Investigation* function is in charge of reviewing the received MR from different ITS-S to corroborate the information that is being reported. Finally, if the investigation is successful the *Response* function is the one in charge of choosing the punishment to be applied to the correspondent ITS-S, such as revocation of all the certificates. The detected misbehaviour agents are added to the *Certificate Revocation List* (CRL) that is shared through the network to make it known to the individual ITS-S.

The three processes described above are modelled in Figure 32, while the list of functional exchanges can be seen in Table 13.

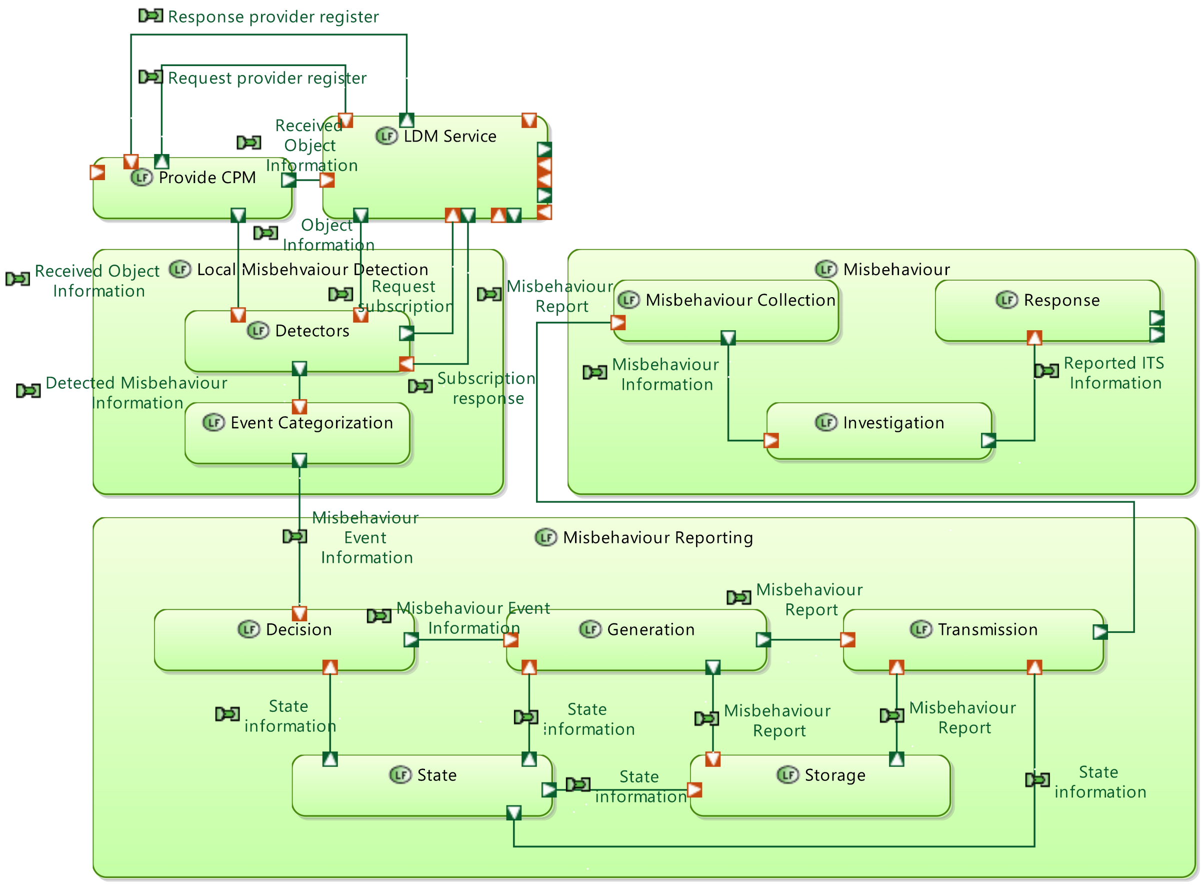


Figure 32: Misbehaviour Detection and Report Functional Data Flow

Table 13: [Misbehaviour Detection and Reporting Functional Exchanges

Functional Exchange	**Source Function**	**Target Function**
Received Object Information	Provide CPM	LDM Service
Request provider register	Provide CPM	LDM Service
Response provider register	LDM Service	Provide CPM
Request subscription	Detectors	LDM Service
Subscription response	LDM Service	Detectors
Object information	LDM Service	Detectors
Received Object Information	Provide CPM	Detectors
Detected Misbehaviour Information	Detectors	Event Categorization
Misbehaviour Event Information	Event Categorization	Decision
Misbehaviour Event Information	Decision	Generation
Misbehaviour Report	Generation	Storage
Misbehaviour Report	Generation	Transmission
Misbehaviour Report	Storage	Transmission
State Information	State	Decision
State Information	State	Generation
State Information	State	Storage
State Information	State	Transmission
Misbehaviour Report	Transmission	Misbehaviour Collection
Misbehaviour Information	Misbehaviour Collection	Investigation
Reported ITS Information	Investigation	Response

5.2.8 Applications

Applications make usage of all the data available in the *Facilities Layer* to achieve specific goals. Although the *Application Layer* is out of the scope of this thesis, the layer and some example functions based on the two main ITS Applications: *Road Hazard Signaling*, specified in ETSI TS 101 539-1 [28], and *Intersection Collision Warning*, from ETSI TS 101 539-2 [20], are created to enable an easier transition if there is the desire to expand on this topic in the future. The two allocated functions to the *Application Layer* can be seen in Figure 33.

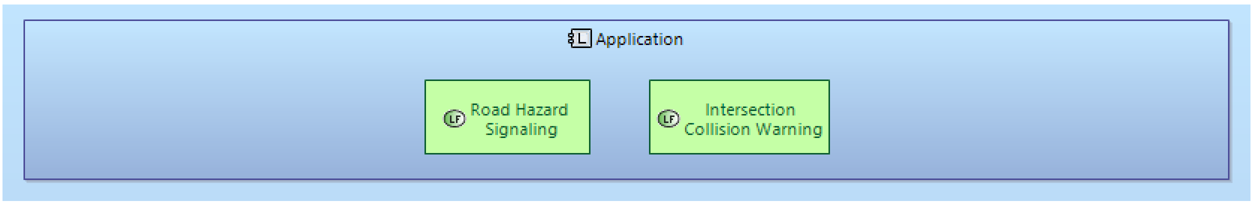


Figure 33: Misbehaviour Detection and Report Data Flow

5.2.9 ITSn Replica

The last step in the Logical Architecture consists of creating a replica of the model created in the system ITS to duplicate it in the external actors *ITSn* and *Infrastructure*, providing them with the ITS functionalities. Same process from Section 5.1.4 is followed, using the REC and RPL tools from Capella to create the replica.

One of the main benefits from an MBSE approach is the reusability that is given to already implemented models, as this replica can be used as many times as desired and is also not constrained to this specific project.

The resulting architecture is shown in Results Section 6.3 in a simplified version, only adding to the diagram the lower levels of the ITS Architecture to obtain a more view, while the complete replica from *ITSn* is added to the Appendice D.

5.3 State Machine

Capella offers the possibility to create State Machine diagrams to showcase the different modes in which the system can operate and what conditions, processes or functions trigger the transition between them. After creating the State Machine, each function in the architecture can be assigned to any number of modes to indicate that it is enabled in that operation scenario. The combination of all the functions available in a mode is named after Configuration.

The implemented State Machine refers to the different perception modes that the system can operate in, to allow the possibility of selecting which are the enabled functions in each configuration. It also illustrates the certificate state in the secure communication network and the steps needed to connect to the network. In the following paragraphs all the states are described:

Figure 34 illustrates the ITS State Machine diagram in Capella. The next paragraphs describe the different modes shown in the State Machine.

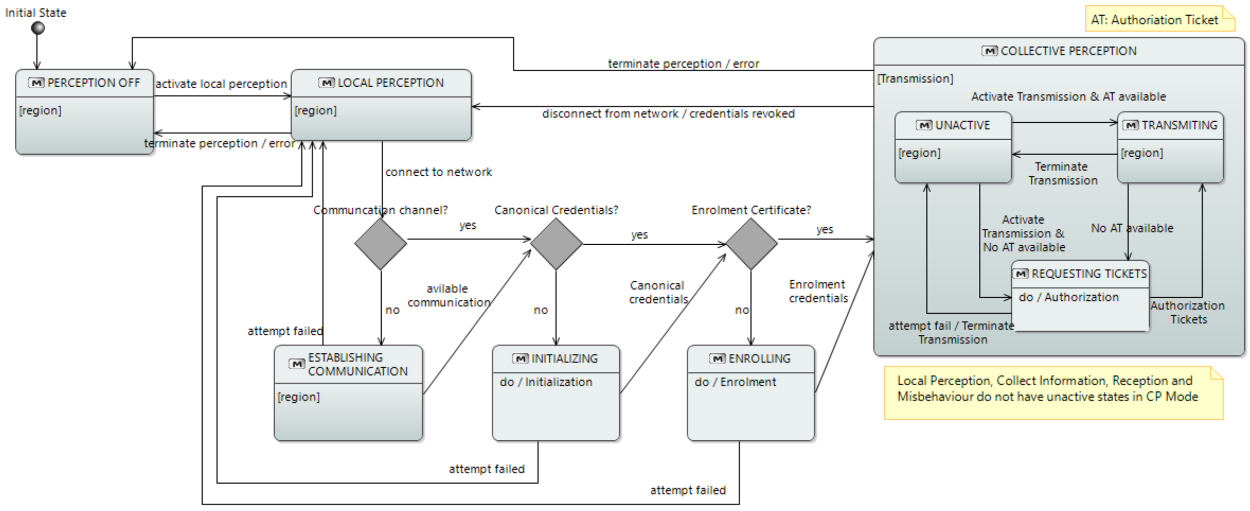


Figure 34: ITS State Machine

PERCEPTION OFF The Initial State of the State Machine is PERCEPTION OFF, where the perception functions are disabled. This mode will also work as the safe mode of the perception system, meaning that if an error is detected the system will transition to it. Transition to LOCAL PERCEPTION is done when the local perception is activated.

LOCAL PERCEPTION The system operates locally in this mode, executing object detection's and collecting information, but without having any exchange of data with other ITS-S. Through the Connect to network transition the COLLECTIVE PERCEPTION mode can be reached, but three checks need to be made: to have a communication channel available, have the Canonical Credentials and possess a valid Enrolment Certificate. If the answer is no to any of the checks the system enters the corresponding mode to obtain the lacking element.

ESTABLISHING COMMUNICATION The system searches for a network to which connect to. In the meantime the functions from LOCAL PERCEPTION are still active. If the attempt fails the system returns to LOCAL PERCEPTION mode, if it succeeds, it advances to the next check to reach COLLECTIVE PERCEPTION.

INITIALIZING The systems follows the procedures to obtain the Canonical Credentials (CC) which are necessary to request the Enrolment Certificate, as seen in Section 5.2.6. This mode will most likely only be entered on the first transition to COLLECTIVE PERCEPTION as the Canonical Credentials do not have a valid time life. After the CC are obtained, the State Machine transitions to the last check. If the attempt is failed it returns to the LOCAL PERCEPTION mode. As in the previous mode, LOCAL PERCEPTION functions are still active during the creation of the credentials.

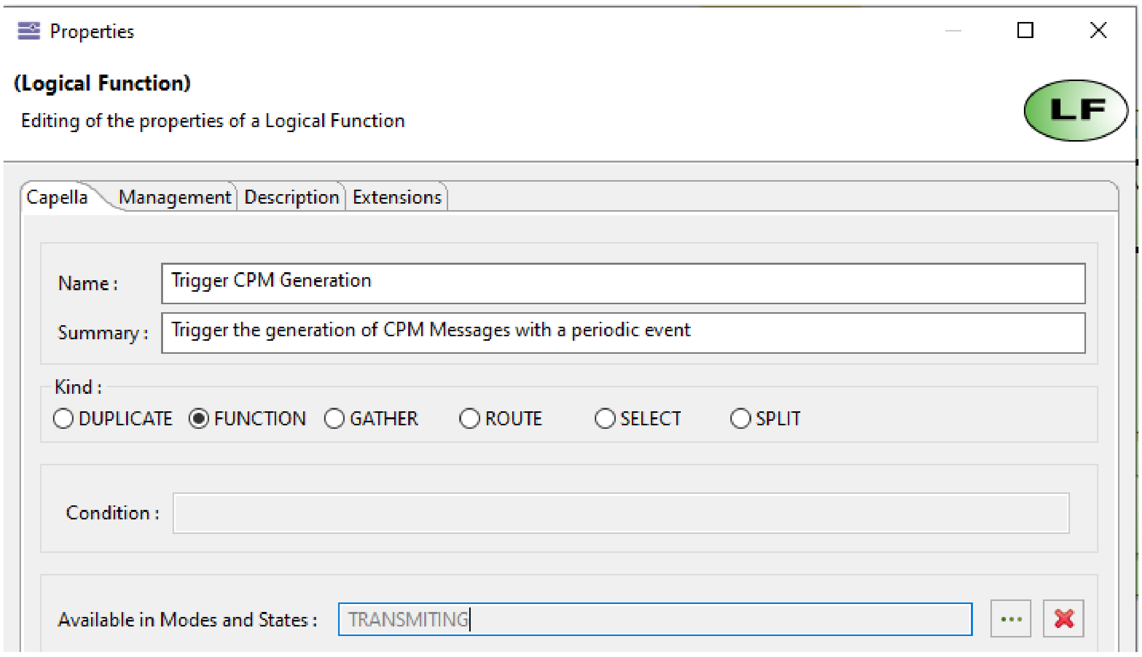
ENROLLING The system creates a request to obtain the Enrolment Credentials (EC) to be granted access to the ITS network. Different from the CC, the EC do have an end of life time where they are no longer valid. Therefore, this mode can be accessed various times during the execution after the credentials are revoked, being the reason time or other factors. If the attempt is failed, the system returns to LOCAL PERCEPTION, while if succeeds it transitions to COLLECTIVE PERCEPTION. The local perception systems remained enabled at this mode too.

COLLECTIVE PERCEPTION The system uses local perception in conjunction with the collective perception, enabling the CP Service functions to transmit and receive object data. Local Perception, Collect Information, CPM reception and Misbehaviour are enabled at all times and do not have any specific configuration inside this mode. However, CPM Transmission can be activated or deactivated by the system and also depends on the availability of Authorization Tickets (AT). Therefore three internal modes are established:

- **UNACTIVE** The transmission operation is deactivated. When the function activate CPM Transmission Operation is called it transitions to TRANSMITTING if there are already Authorization Tickets (AT) available. If that is not the case and the transmission is activated, the mode REQUESTING TICKETS is entered.
- **REQUESTING TICKETS** Function Authorization is called to request AT for the CPM Transmission. When the AT are obtained, the mode TRANSMITTING is entered. The system returns to UNACTIVE if the attempt to obtain AT fails or if the transmission is terminated during the request.
- **TRANSMITTING** The process of CPM Generation and Transmission is enabled. When the AT run out, the system transitions to REQUESTING TICKETS to obtain new ones. If the operation is terminated the system returns to UNACTIVE.

The mode COLLECTIVE PERCEPTION is exited when the system disconnects from the network or the credentials are revoked by any cause, entering the LOCAL PERCEPTION mode. If the perception is terminated completely or an error is detected the transition occurs to PERCEPTION OFF instead.

As introduced above, the implementation of a State Machine allows the user to assign specific Modes to a function where it is enabled, creating Configurations. This information has been added to the implemented Capella model, and an example is shown in Figure 35 where Trigger CPM Generation only shows as Available in TRANSMITTING mode, while in Figure 37 the Process Perception Data is available in all the modes except of PERCEPTION OFF.


Properties

(Logical Function)

Editing of the properties of a Logical Function

Name : Trigger CPM Generation

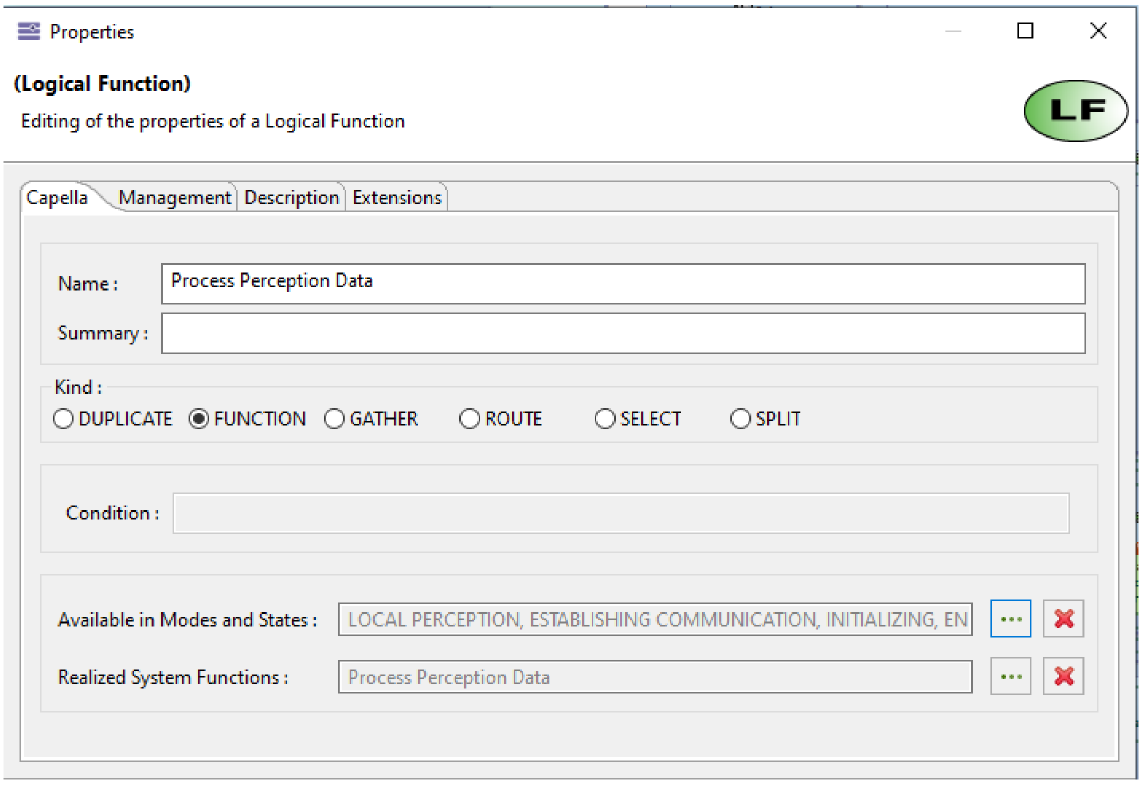
Summary : Trigger the generation of CPM Messages with a periodic event

Kind : DUPLICATE [✓ FUNCTION] GATHER ROUTE SELECT SPLIT

Condition :

Available in Modes and States : TRANSMITING

Figure 35: Trigger CPM Generation Available Modes


Properties

(Logical Function)

Editing of the properties of a Logical Function

Name : Process Perception Data

Summary :

Kind : DUPLICATE [✓ FUNCTION] GATHER ROUTE SELECT SPLIT

Condition :

Available in Modes and States : LOCAL PERCEPTION, ESTABLISHING COMMUNICATION, INITIALIZING, EN

Realized System Functions : Process Perception Data

Figure 36: Process Perception Data Available Modes

6 Results

This section covers the final results of the implemented *Secure Collective Perception Service in an ITS Environment* Capella model.

Section 6.1 shows the final System Analysis Architecture with the added RPL to ITSn and Interface. Additional functional exchanges *Transmitted CPM* have been added to model the messages going through the network after leaving the ITS-S.

Section 6.2 contains the final Logical Function Breakdown from the Logical Architecture Level after all the subfunctions from Sections 5.2.2 to 5.2.7 are added to the model. The relationship between the first parent functions and the specific leaf functions is clearly appreciated. Important to notice how the lower a function in the hierarchy, the more specific in the actual description of the process. Therefore, the upper layers describe *What* the system does, while the lower ones transcribe this *What* into *How*.

Section 6.3 shows the final Logical Architecture Diagram containing the ITS Layers in the form of Components and the respective allocated functions. Notice how the functions shown in the diagram are not all Leaf functions. Just as an example, *CPM Generation* and Manage CPM Transmission are located in the diagram, but their respective subfunctions are introduced in Section 5.2.4. This illustrates the power of encapsulation of a MBSE approach where the user can create understandable diagrams that are not too complex to follow, while at the same time create deeper specification hidden inside other function blocks. As done with the System Analysis, *Transmitted CPM* functional exchanges are added to model messages being exchanged in the network between ITS-S.

6.1 Final System Analysis Architecture

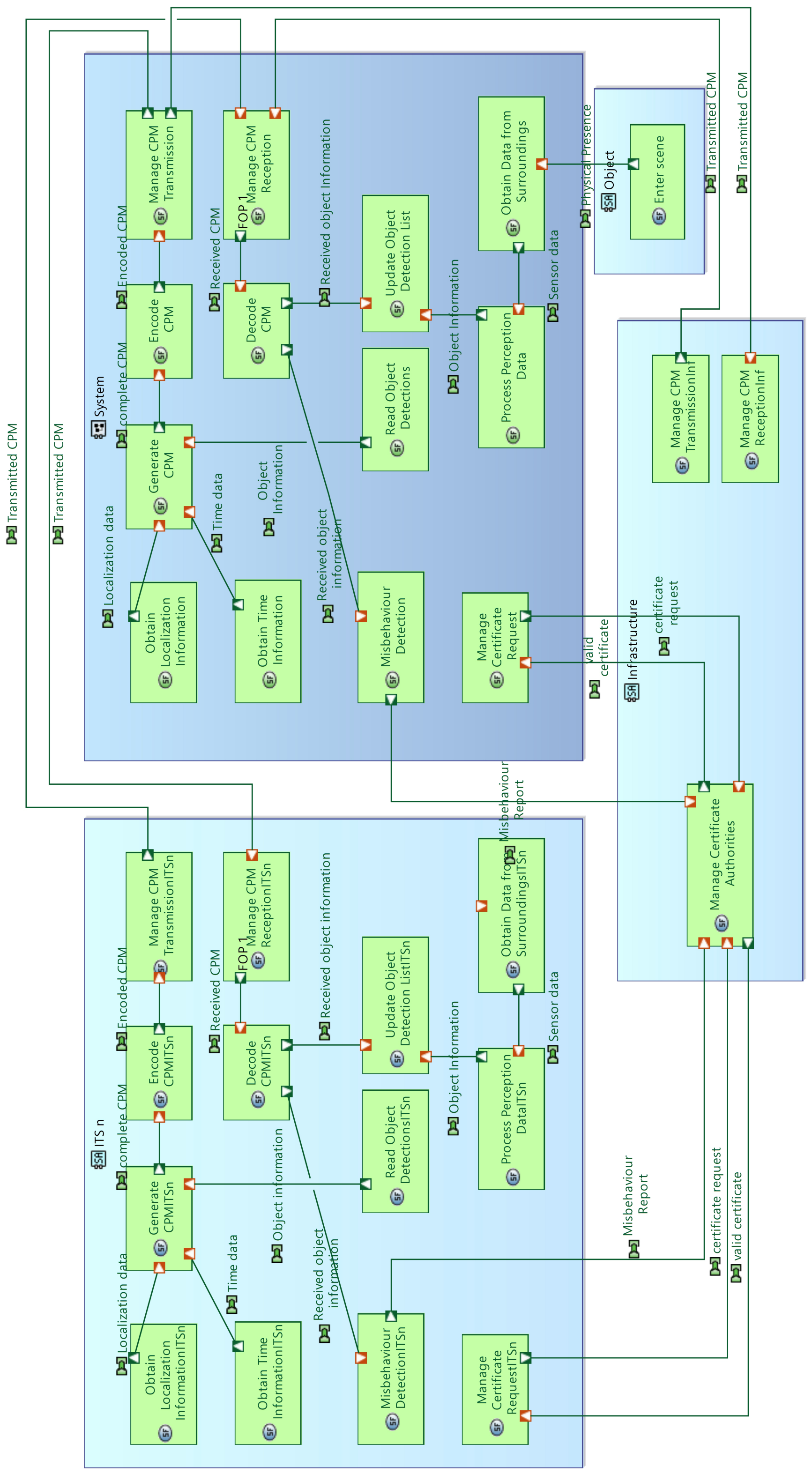


Figure 37: System Analysis Final Architecture

6.2 Logical Functional Breakdown

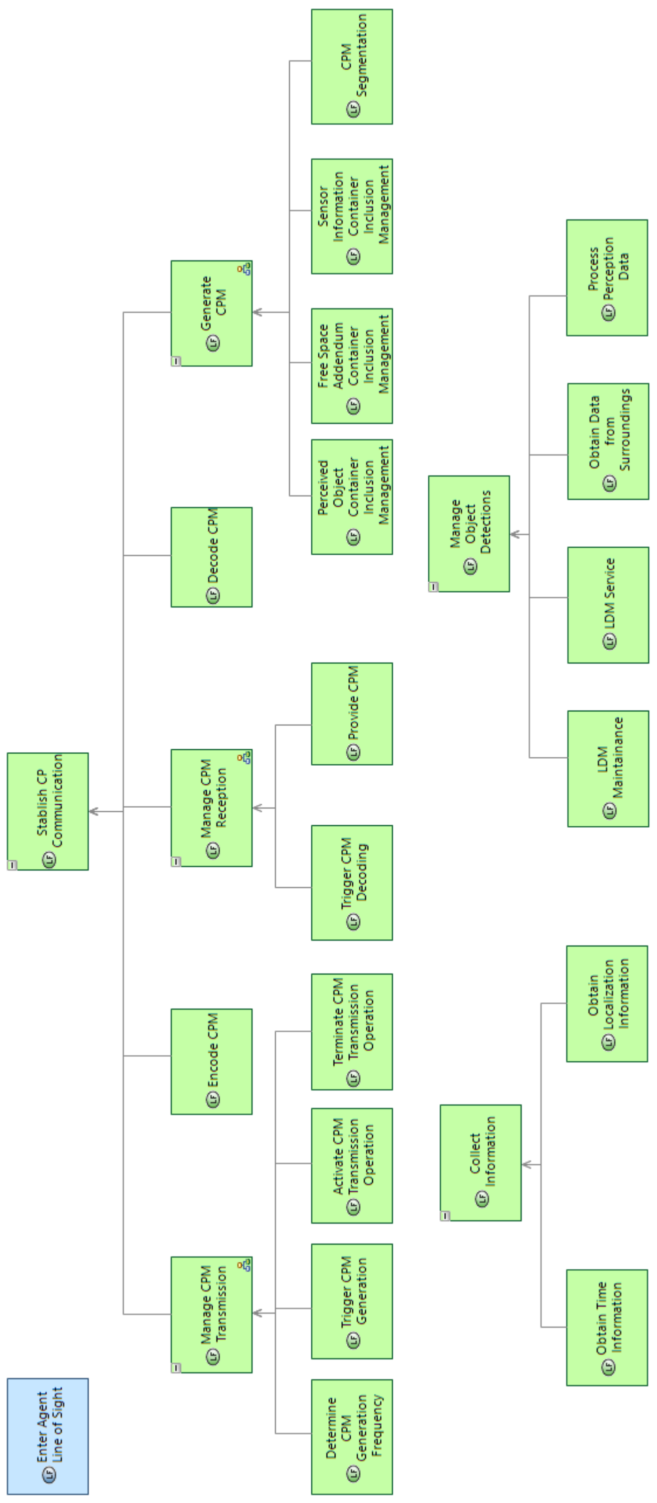

Stablish CP Communication
Enter Agent Line of Sight
Manage CPM Transmission
Encode CPM
Manage CPM Reception
Decode CPM
Generate CPM
Determine CPM Generation Frequency
Trigger CPM Generation
Activate CPM Transmission Operation
Terminate CPM Transmission Operation
Trigger CPM Decoding
Provide CPM
Perceived Object Container Inclusion Management
Free Space Addendum Container Inclusion Management
Sensor Information Container Inclusion Management
CPM Segmentation
Collect Information
Obtain Time Information
Obtain Localization Information
Manage Object Detections
LDM Maintainance
LDM Service
Obtain Data from Surroundings
Process Perception Data

Figure 38: Logical Functional Breakdown Part 1

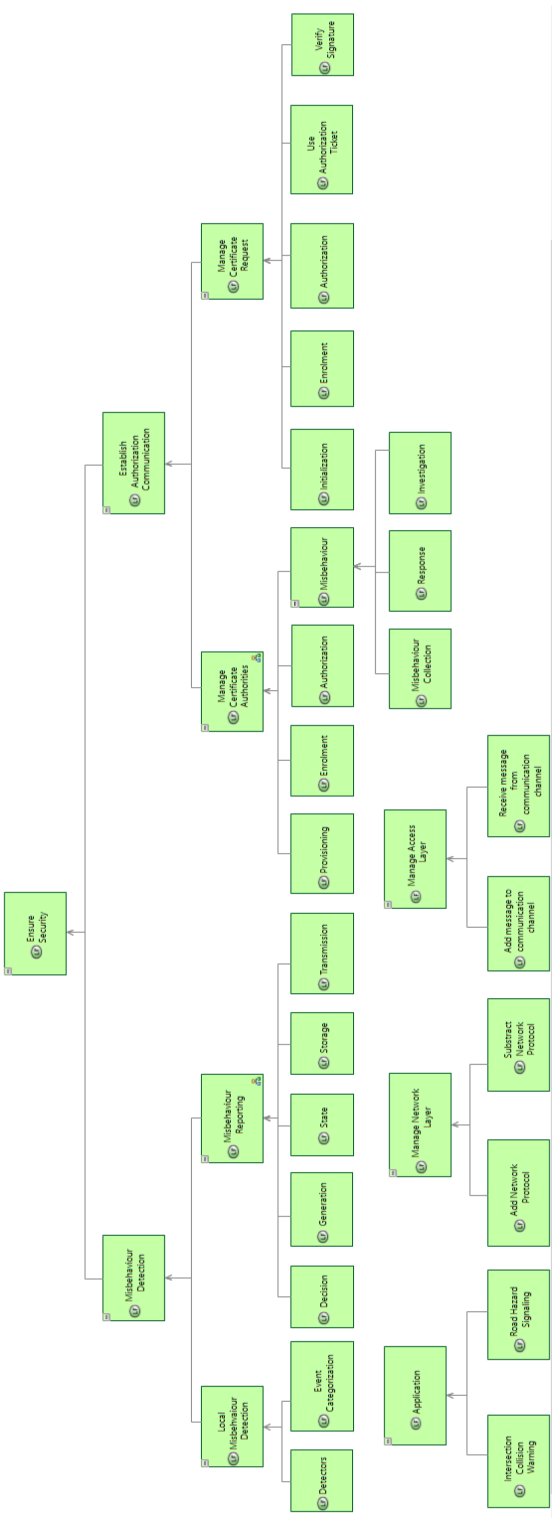


Figure 39: Logical Functional Breakdown Part 2

6.3 Final Logical Architecture

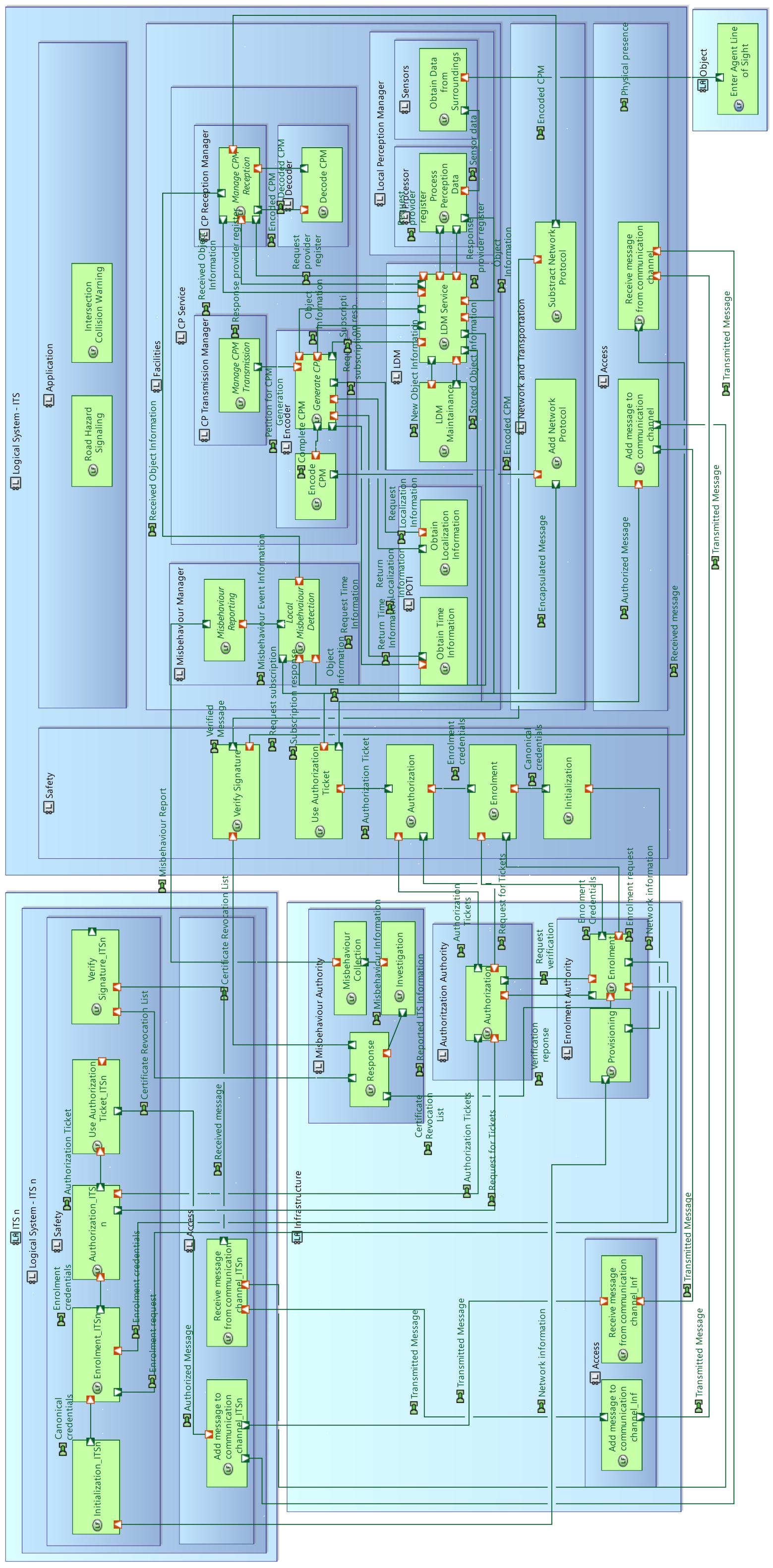


Figure 40: Logical Architecture Final Architecture

7 Conclusion and Future Developments

Enhancing Autonomous Driving Systems with Collective Perception, currently under standardization, has a lot of potential and is increasingly attracting attention as a solution to the physical limitation of vehicles on-board sensors packages.

This master thesis has modelled an architecture using the open-source MBSE software Arcadia Capella and following the development workflow proposed by the Arcadia Method. In addition, the developed model is compliant with an extensive collection of the European Telecommunications Standards Institute (ETSI) standards from the Intelligent Transport Systems (ITS) family.

Through the different architectural levels of abstraction, the benefits of applying a Systems Engineering approach in engineering projects has been proved. Firstly, a high level of reusability of the architectural components has been achieved, specially with multiple instantiation of the ITS Stations (ITS-S). Secondly, it has demonstrated how a complex system, coming from various research papers and extensive ETSI standards, is decomposed into subsystems, easier to understand and implement. This allows a smooth transition from the design phase to the implementation of the project. Thirdly, relevant information is placed at different levels of abstraction of the model to specify future implementation requirements. For example, reference to technical requirements of the standard are placed inside the function description, or functions availability according operation modes from the State Machine.

The developed Collective Perception Architecture integrates smoothly with the architectural levels specified for Smart Cities, as it aligns with the layer distribution of an ITS. The challenges of Heterogeneity, Data Collection and Security, are all addressed in the proposed architecture, based on standards that can adapt to any specific configuration, managing the collection and storage of data, and implementing various processes to improve security.

This thesis can be further developed in various directions. A first extension would be to design a Physical Architecture with the goal to have a specific deployment of the project. A second development would be to implement the internal behaviour of the modelled functions of the system. This would allow to perform simulations to study the robustness of the network. Finally, subcomponents of the modelled Collective Perception Architecture can be further developed as a new Capella project. The most interesting subsystems in this regard, from the perception point of view, would be the sensor data acquisition and sensor fusion processing. Concerning the communication, modelling the communication channel would erase the assumption of an ideal message exchange. Exploring how Misbehaviour Detection algorithms can be modelled is a very promising future research topic.

References

[1] Murad Khan Bhagya Nathali Silva and Kijun Han. Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities. *Sustainable Cities and Society*, 38:697–713, 2018.

[2] E. Poll A. Serban and J. Visser. A standard driven software architecture for fully autonomous vehicles. *IEEE International Conference on Software Architecture Companion (ICSA-C)*, 2018.

[3] Road vehicles — safety and cybersecurity for automated driving systems — design, verification and validation, January 2020.

[4] Etsi tr 102 638 v1.1.1: Intelligent transport systems (its); vehicular communications; basic set of applications; definitions.

[5] Etsi en 302 665 v1.1.1 intelligent transport systems (its); communications architecture.

[6] Etsi ts 103 324: Intelligent transport system (its); vehicular communications; basic set of applications; specification of the collective perception service.

[7] Iris Graessler and Julian Hentze. The new v-model of vdi 2206 and its validation. 68(5):312–324.

[8] Eclipse capella.

[9] Etsi tr 103 460: Intelligent transport systems (its); security; pre-standardization study on misbehaviour detection; release 2.

[10] Ferdin Muetsch Martin Boheme, Marco Stang and Eric Sax. Talkycars: A distributed software platform for cooperative perception.

[11] P. Kulkarni and T. Farnham. Smart city wireless connectivity considerations and cost analysis: Lessons learnt from smart water case studies. *IEEE Access*, 4:660–672, 2016.

[12] A. Bashar N. Mohammad, S. Muhammad and M. A. Khan. Formal analysis of human-assisted smart city emergency services. *IEEE Access*, 7:60376–60388, 2019.

[13] P. Wilson. State of smart cities in u.k. and beyond. *IET Smart Cities*, 1(1):19–22, 2019.

[14] W. G. Rong Z. Xiong, H. Sheng and D. E. Cooper. Intelligent transportation systems for smart cities: A progress review. *Science China*, 55(12):2908–2914, 2012.

[15] Sanjay Kumar Sharma Anand Nayyar Anand Paul Tarana Singh, Arun Solanki. A decade review on smart cities: Paradigms, challenges and opportunities. *IEEE Access*, 10:68319–68364, 2022.

[16] Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles.

[17] Philip Koopman and Michael Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 9:90–96, 2017.

[18] V2x communications message set dictionary j2735.

[19] Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service.

[20] Etsi ts 101 539-1: Intelligent transport systems (its); v2x applications; part 2: Intersection collision risk warning (icrw) application requirements specification.

[21] Jonathan Petit Cong Chen Mohammad Raashid Ansari, Jean-Philippe Monteuuis. V2x misbehavior and collective perception service: Considerations for standardization.

[22] Etsi en 302 890-2 intelligent transport systems (its); facilities layer function; part 2: Position and time management (poti); release 2.

[23] Etsi en 302 636-5-1 v2.1.0: Intelligent transport systems (its); vehicular communications; geonetworking; part 5: Transport protocols; sub-part 1: Basic transport protocol.

[24] Etsi ts 103 248 v2.1.1: Intelligent transport systems (its); geonetworking; port numbers for the basic transport protocol (btp); release 2.

[25] Etsi ts 102 940 v2.1.1: Intelligent transport systems (its); security; its communications security architecture and security management; release 2 s.

[26] Iso/iec/ieee 15288:2015 systems and software engineering — system life cycle processes.

[27] Barcelo J. Ramos A. L., Ferreira J. V. Model-based systems engineering: An emerging approach for modern systems.

[28] Etsi ts 101 539-1: Intelligent transport systems (its); v2x applications; part 1: Road hazard signalling (rhs) application requirements specification.

[29] Etsi ts 102 894-1 v1.1.1: Intelligent transport systems (its); users and applications requirements; part 1: Facility layer structure, functional requirements and specifications.

[30] Etsi ts 103 759 v2.1.1:intelligent transport systems (its); security; misbehaviour reporting service; release 2.

[31] Intelligent transport systems (its); vehicular communications; basic set of applications; local dynamic map (ldm).

[32] Robert Hammett. Design by extrapolation: an evaluation of fault-tolerant avionics. *IEEE*, 2001.

[33] John K. Tsotsos Amir Rasouli. Autonomous vehicles that interact with pedestrians: A survey of theory and practice. *IEEE*, 2018.

Appendices

A ISO/TR 4808 Road vehicles - FS_2 and FS_6 Capabilities

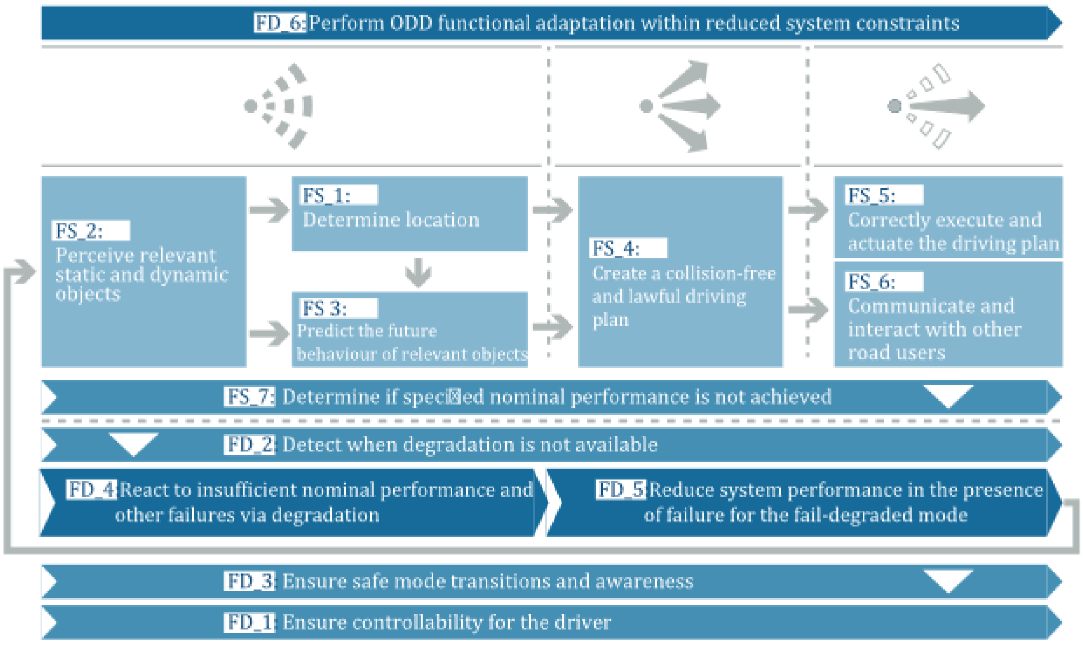

FD_6: Perform ODD functional adaptation within reduced system constraints
FS_2: Perceive relevant static and dynamic objects
FS_1: Determine location
FS 3: Predict the future behaviour of relevant objects
FS_4: Create a collision-free and lawful driving plan
FS_5: Correctly execute and actuate the driving plan
FS_6: Communicate and interact with other road users
FS_7: Determine if specified nominal performance is not achieved
FD_2: Detect when degradation is not available
FD_4: React to insufficient nominal performance and other failures via degradation
FD_5: Reduce system performance in the presence of failure for the fail-degraded mode
FD_3: Ensure safe mode transitions and awareness
FD_1: Ensure controllability for the driver

Figure 41: Realizing fail-safe and fail-degraded capabilities[3]

FS_2: Perceive relevant static and dynamic objects An automated driving system uses sensors to perceive its environment and safely navigate roads and highways. It gives priority to entities that pose the greatest risk of collision, such as other vehicles, obstacles, and road hazards. These entities are perceived, pre-processed, and provided to the system in a safe manner. Examples of entities that the system may account for include dynamic objects, such as other vehicles and their movement characteristics, static objects, such as road boundaries and traffic signals, and obstacles that exceed a certain size.

Sensor fusion is the core element of an automated driving perception system, as it allows the system to integrate data from a variety of onboard sensors, such as cameras, radar, and lidar, localization, ego-motion and V2X information. As shown in Figure 42, this data is used to generate a model of the present environment, which the system uses for interpretation, prediction, and planning. Traffic rules may be used to optimize the world model and ensure safe operation of the vehicle. The semantic knowledge of the perceived environment is important for the system to understand the objects and entities it has detected and their locations.

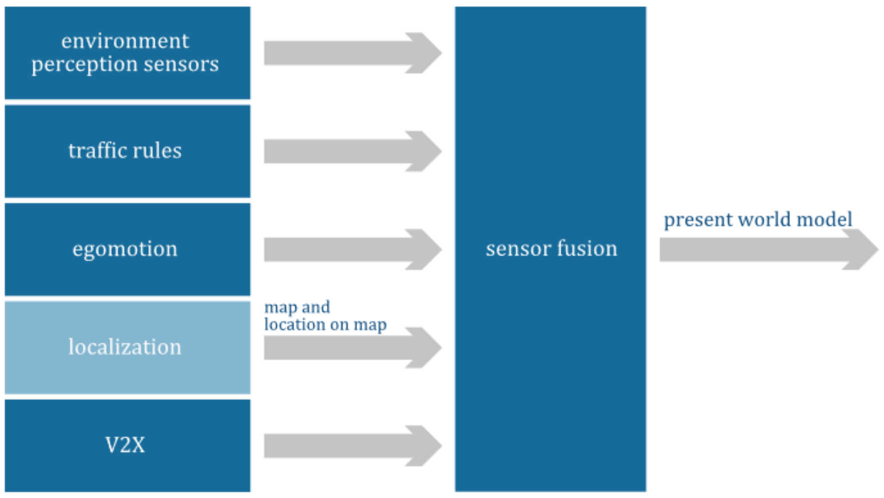


Figure 42: FS_2: Perceive relevant static and dynamic objects[3]

FS_6 Communicate and interact with other road users The self-driving vehicle moves in a predictable way, following all traffic laws and signals. This includes using turn signals before changing lanes and maintaining a safe distance from other vehicles. The vehicle may also use visual and auditory signals, as well as communication technologies like V2X, to communicate its intentions and actions to other road users (see Figure 43).

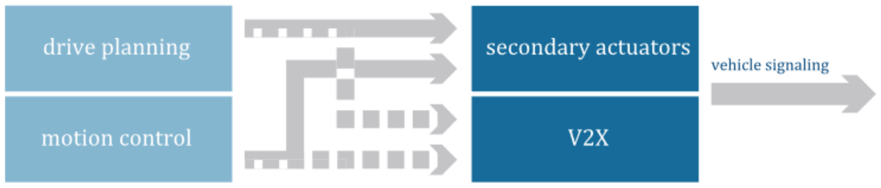


Figure 43: FS_6: Communicate and interact with other road users[3]

B Automated Driving System Challenges

Authors in [17] explore the different range of safety challenges and concerns that Automated Driving Systems face:

Safety Engineering Assuming the existence of small-scale deployment of fully autonomous Level 4 vehicles on the road, the challenge becomes managing failures that may be infrequent for any single vehicle, but will happen often when deploying at a big scale. The current accepted practice for vehicle computer-based system safety, which relies on the driver to be responsible for vehicle safety, needs to be modified as a fully autonomous vehicle eliminates the human driver's responsibility entirely. Another safety certification concern is validating the use of self-adaptive system behavior by autonomous vehicles, which may lead to different behaviors during operation than was displayed during testing and certification.

One method for ensuring safety in highly autonomous systems is to switch control back to a human driver in the event of an equipment failure, allowing them to take over driving responsibility. However, if the human is not paying attention, there needs to be a fail-operational autonomy capability in place to maintain control until the human can take over. Cars can achieve a safe state quickly, so a strategy is to switch to a short "safing mission" mode when a critical component fails, allowing the car to pull over to the side of the road without the need for complex driving autonomy. This "safing mission" has less strict reliability and redundancy requirements and potentially reduce overall system cost and complexity, while still keeping the vehicle safe. By designing a safe shutdown mission capability, the safety requirements on primary vehicle autonomy can also be relaxed.

Social Acceptance The process of gaining social acceptance for autonomous vehicles will undoubtedly be complex. While a primary incentive for adoption is the expectation that autonomous vehicles will be safer drivers than humans, it is unrealistic to assume that this will mean zero accidents, especially in the early stages of implementation. Some cases will be straightforward, such as when avoiding a collision is physically impossible, such as when a tree falls on top of a car during a storm. However, determining the standard for autonomous safety will be challenging, such as whether it should be better than an excellent human driver or just a typical one, and how a typical driver should be characterized. Particularly tricky situations will arise when an ordinary human driver could have avoided an accident, but the autonomous vehicle did not, leading to questions about the vehicle's liability in such cases.

Computing Hardware Developing ultra-cheap hardware with safe failure behaviors is certainly not an easy task, even with the usage of a safe mission plan. To do this, a well implemented hardware and software architecture is required. As an example of progress, chip manufacturers have created dual-core processing circuits that can at least partially redundancy for computations [32]. To provide enough redundancy and fault tolerance, though, a more extensive strategy will probably be needed. Such systems have been designed for aircraft and other safety-critical applications for a very long time, which provides extensive background knowledge.

The difficulty of latent fault detection in hardware is, although more subtle, also an important topic. Reliability calculations are routinely performed at the start of each mission under the assumption of completely fault-free redundancy. The advantages of redundancy are drastically reduced by any undiscovered flaw. Self-diagnosis errors of even a few percentage points have significant effects on the level of achievable dependability. For instance, obtaining only 95% test coverage might drastically lower the level of feasible reliability for redundant automobile systems.

Due to the fact that undiagnosed failures can build up over the course of a vehicle's working life, the likelihood of experiencing multiple independent, undiagnosable failures is much higher than the likelihood of multiple failures occurring during a single driving mission for diagnosed parts of the system. Therefore, it will be crucial to design chips with a very high level of diagnostic coverage that can be self-tested before each driving cycle.

Human Computer Interaction As self-driving cars become more prevalent, their ability to communicate and work together with humans will become increasingly important. It is evident that there are risks associated with human supervisors being inattentive in systems that are nearly fully autonomous. Even fully autonomous vehicles will need to ensure that their passengers feel safe, in order to build trust and confidence, and will need to learn how to anticipate the actions of other vehicles. While some argue that customer trust is not a safety concern, it is crucial for the adoption of this technology, which, if it fulfills its promise of saving lives on the road, will indirectly impact safety [33].

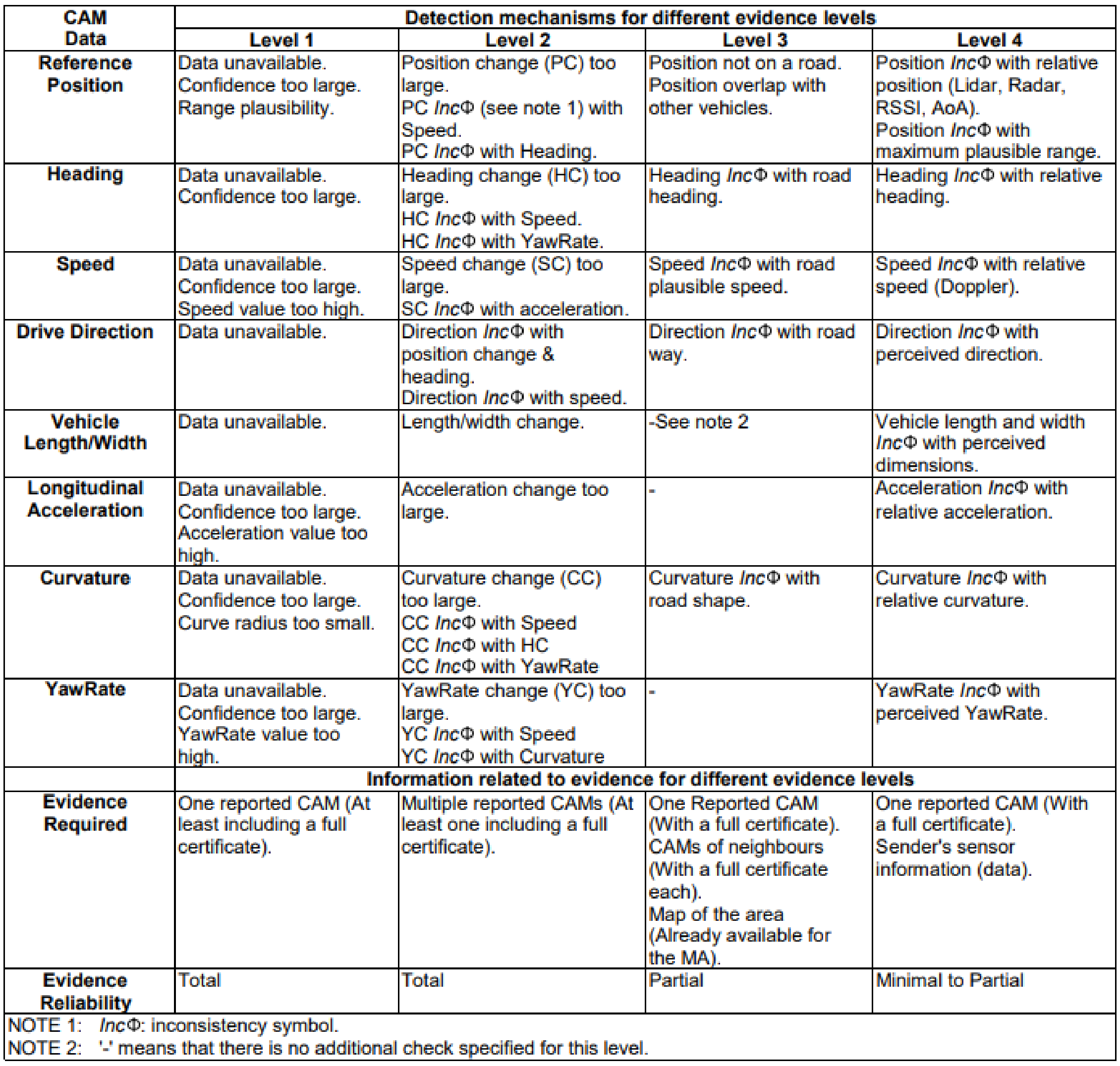
Many areas will require automation to behave in a way that humans can understand. This means that humans should be able to easily comprehend the automation's intentions and why it exhibited a particular behavior. This will be particularly important in fields such as human-vehicle interaction safety, where it is necessary to know if the autonomous vehicle has noticed a pedestrian, for example, and in testing coverage, where it is necessary to determine whether the automation has correctly identified the reason for its actions. Additionally, design comprehension will be important in understanding where the machine learning system is located and whether it covers all relevant aspects or just a subset of them [33].

Legal Dealing with legal liability will be a significant challenge in the early stages of deploying autonomous vehicles. When a mishap occurs involving a fully autonomous vehicle, the passenger may not be at fault, as they are justifiably inattentive or even asleep. In such situations, the primary source of information about the mishap would be the vehicle's data logs. However, this data cannot be blindly trusted if the mishap occurred due to a malfunction in the vehicle's system. In such cases, it cannot be assumed that the data collected by the malfunctioning system is accurate. To address this issue, an independent data recording system might need to be intentionally designed to provide reliable data for mishap forensics. The currently deployed Event Data Recording devices may need to be reevaluated and reconfigured to ensure that they can provide adequate data for this purpose.

A critical issue regarding liability will be determining who is ultimately responsible for ensuring proper vehicle operation. This responsibility falls on the vehicle occupant who chose to rent a car with noticeable sensor damage, the vehicle manufacturer who relied on a faulty training dataset from a third party, the mechanic who installed an incompatible replacement sensor software, the mapping update service that failed to record a bridge washout, or the operating system vendor who failed to deploy a security patch in a timely manner to prevent a malicious incident. Although some of these issues are purely legal, resolving many legal questions will require a solid technological foundation. While much experience is being gained from pilot deployments, the legal implications of autonomous vehicles are still largely unknown, along with several other legal topics.

C Misbehaviour Detection Strategies

Table 44 presents potential local misbehaviour detection mechanisms related to the content of the received CAMs, i.e. CAM information components with suspicious values, and related evidence issues for four different evidence levels [9].



CAM Data	Detection mechanisms for different evidence levels			
	Level 1	Level 2	Level 3	Level 4
Reference Position	Data unavailable. Confidence too large. Range plausibility.	Position change (PC) too large. PC *Inc*Φ (see note 1) with Speed. PC *Inc*Φ with Heading.	Position not on a road. Position overlap with other vehicles.	Position *Inc*Φ with relative position (Lidar, Radar, RSSI, AoA). Position *Inc*Φ with maximum plausible range.
Heading	Data unavailable. Confidence too large.	Heading change (HC) too large. HC *Inc*Φ with Speed. HC *Inc*Φ with YawRate.	Heading *Inc*Φ with road heading.	Heading *Inc*Φ with relative heading.
Speed	Data unavailable. Confidence too large. Speed value too high.	Speed change (SC) too large. SC *Inc*Φ with acceleration.	Speed *Inc*Φ with road plausible speed.	Speed *Inc*Φ with relative speed (Doppler).
Drive Direction	Data unavailable.	Direction *Inc*Φ with position change & heading. Direction *Inc*Φ with speed.	Direction *Inc*Φ with road way.	Direction *Inc*Φ with perceived direction.
Vehicle Length/Width	Data unavailable.	Length/width change.	-See note 2	Vehicle length and width *Inc*Φ with perceived dimensions.
Longitudinal Acceleration	Data unavailable. Confidence too large. Acceleration value too high.	Acceleration change too large.	-	Acceleration *Inc*Φ with relative acceleration.
Curvature	Data unavailable. Confidence too large. Curve radius too small.	Curvature change (CC) too large. CC *Inc*Φ with Speed CC *Inc*Φ with HC CC *Inc*Φ with YawRate	Curvature *Inc*Φ with road shape.	Curvature *Inc*Φ with relative curvature.
YawRate	Data unavailable. Confidence too large. YawRate value too high.	YawRate change (YC) too large. YC *Inc*Φ with Speed YC *Inc*Φ with Curvature	-	YawRate *Inc*Φ with perceived YawRate.
	Information related to evidence for different evidence levels			
Evidence Required	One reported CAM (At least including a full certificate).	Multiple reported CAMs (At least one including a full certificate).	One Reported CAM (With a full certificate). CAMs of neighbours (With a full certificate each). Map of the area (Already available for the MA).	One reported CAM (With a full certificate). Sender's sensor information (data).
Evidence Reliability	Total	Total	Partial	Minimal to Partial
NOTE 1: *Inc*Φ: inconsistency symbol. NOTE 2: '-' means that there is no additional check specified for this level.				

Figure 44: Misbehaviour detection mechanisms for CAMs [9]

Detection levels are defined as follows:

- Level 1: Implausibilities within a single message.
- Level 2: Inconsistencies between successive messages.
- Level 3: Inconsistencies with the local environment.
- Level 4: Inconsistencies with respect to perceived physical attributes (based on-board sensors or V2X physical layer techniques).

D ITSn Replica Logical Architecture

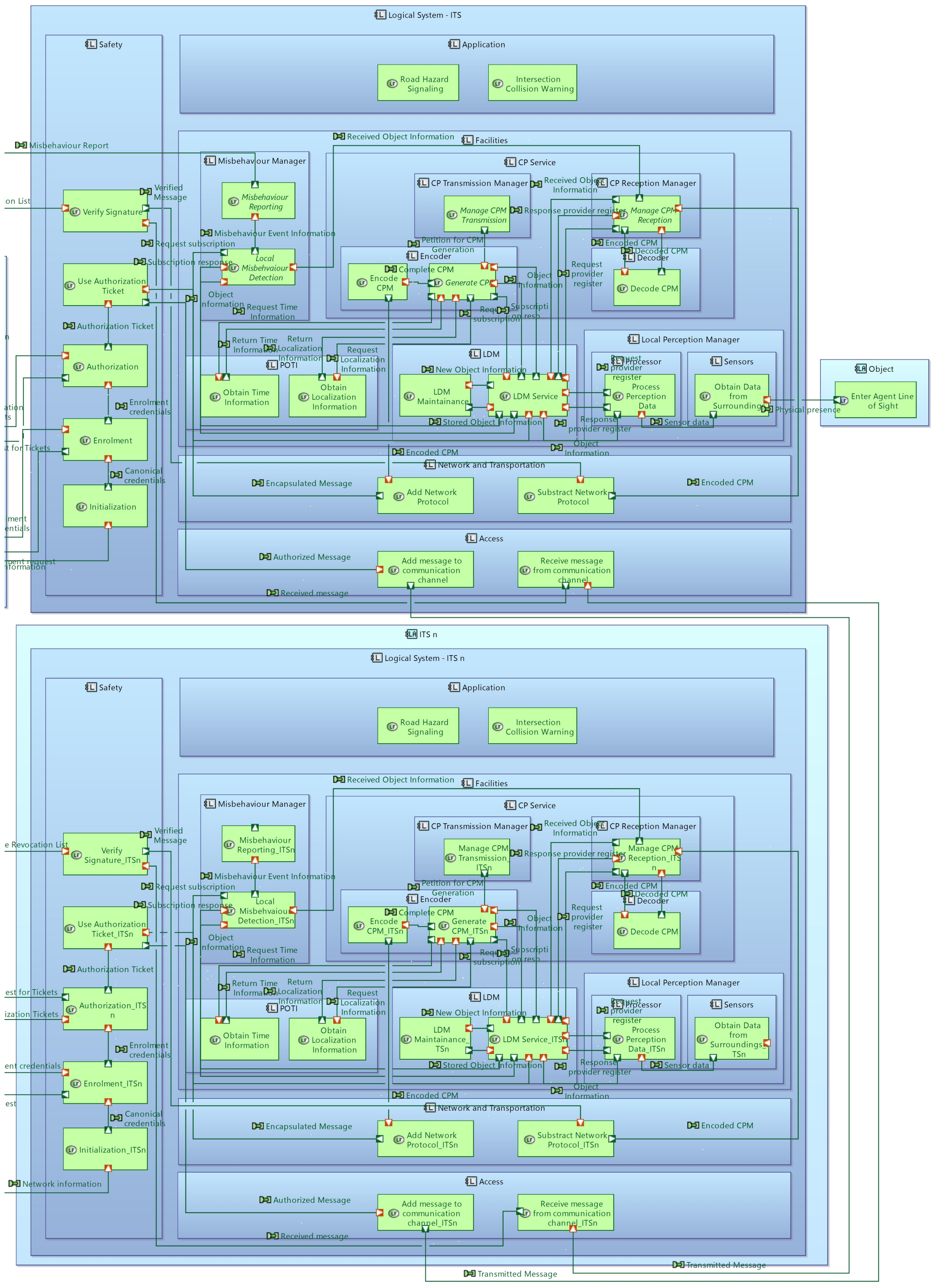
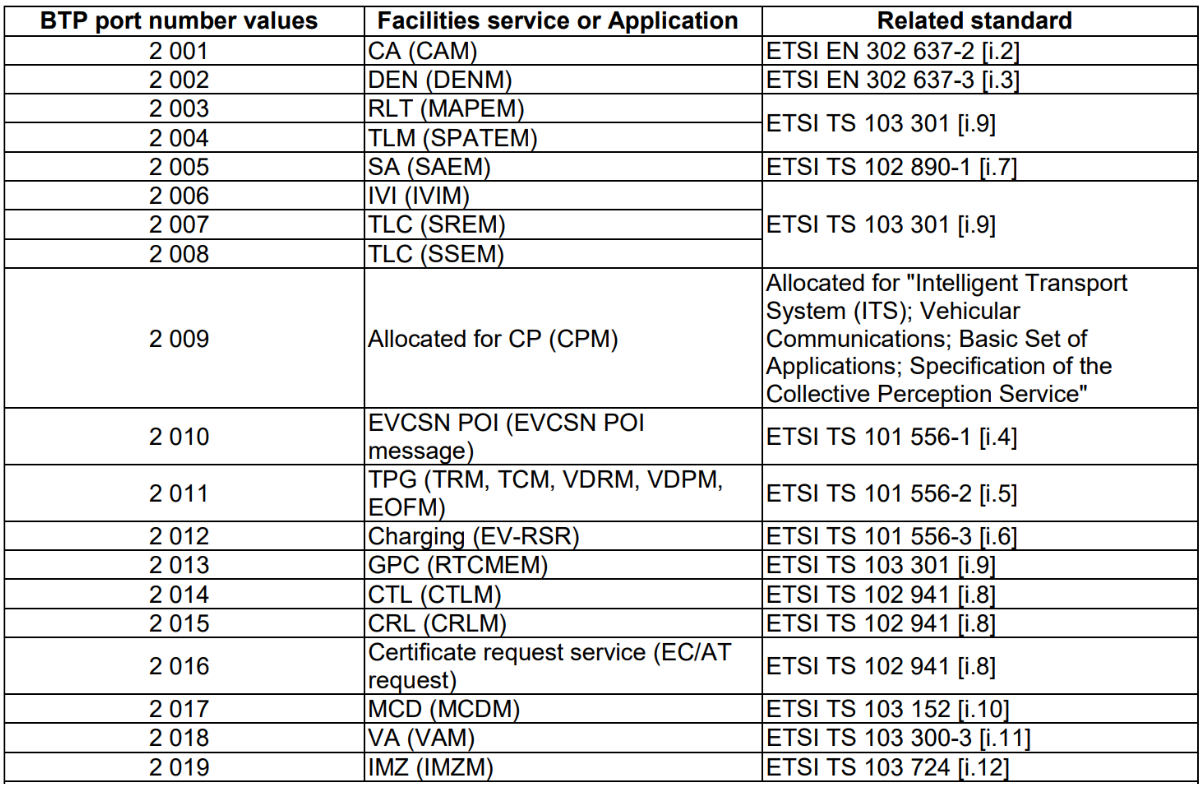


Figure 45: ITSn RPL Logical Architecture

E BTP Port Numbers



BTP port number values	Facilities service or Application	Related standard
2 001	CA (CAM)	ETSI EN 302 637-2 [i.2]
2 002	DEN (DENM)	ETSI EN 302 637-3 [i.3]
2 003	RLT (MAPEM)	ETSI TS 103 301 [i.9]
2 004	TLM (SPATEM)	
2 005	SA (SAEM)	ETSI TS 102 890-1 [i.7]
2 006	IVI (IVIM)	ETSI TS 103 301 [i.9]
2 007	TLC (SREM)	
2 008	TLC (SSEM)	
2 009	Allocated for CP (CPM)	Allocated for "Intelligent Transport System (ITS); Vehicular Communications; Basic Set of Applications; Specification of the Collective Perception Service"
2 010	EVCSN POI (EVCSN POI message)	ETSI TS 101 556-1 [i.4]
2 011	TPG (TRM, TCM, VDRM, VDPM, EOFM)	ETSI TS 101 556-2 [i.5]
2 012	Charging (EV-RSR)	ETSI TS 101 556-3 [i.6]
2 013	GPC (RTCMEM)	ETSI TS 103 301 [i.9]
2 014	CTL (CTLM)	ETSI TS 102 941 [i.8]
2 015	CRL (CRLM)	ETSI TS 102 941 [i.8]
2 016	Certificate request service (EC/AT request)	ETSI TS 102 941 [i.8]
2 017	MCD (MCDM)	ETSI TS 103 152 [i.10]
2 018	VA (VAM)	ETSI TS 103 300-3 [i.11]
2 019	IMZ (IMZM)	ETSI TS 103 724 [i.12]

Figure 46: Well-known BTP Port Numbers