Accurate

Next Generation Positioning On Board Unit

European Global Navigation Satellite Systems Agency (GSA)
Fundamental Elements

Grant Agreement GSA/GRANT/03/2018

Preliminary OBU Architecture

Deliverable no.	D3.1
Dissemination level	Public
Work Package no.	WP3
Main author(s)	Oihana Otaegui (VIC), Marcos Nieto (VIC), Markus Bach (Valeo DE), Matthias Wellens (Valeo CTA)
Co-author(s)	Alexandre Allien (FDC), Lance de Groot (Hexagon), Brett Kruger (Hexagon), Phil Gibeault (Hexagon), Marc Delauney (FDC), Kevin Doherty (Hexagon), Andreas Niemann (Hexagon), Gwenaël Dunand (Intempora)
Version Nr (F: final, D: draft)	F
File Name	D3.1_Preliminary_OBU_architecture
Project Start Date and Duration	01 September 2020, 24 months

Project funded by the Fundamental Elements EU R&D Programme by the GSA

European
Global Navigation
Satellite Systems
Agency

Fundamental
Elements

Document Control Sheet

Main author(s) or editor(s): Oihana Otaegui (VIC), Marcos Nieto (VIC), Markus Bach (Valeo DE), Matthias Wellens (Valeo CTA)

Work area: WP3

Document title: Preliminary OBU architecture

Version history:

Version number	Date	Main author	Summary of changes
v0.1	29.01.2021	Oihana Otaegui (VIC), Marcos Nieto (VIC)	Table of contents
v0.2	26.02.2021	Oihana Otaegui (VIC), Marcos Nieto (VIC), Markus Bach (Valeo DE), Matthias Wellens (Valeo CTA), Alexandre Allien (FDC), Lance de Groot (Hexagon), Marc Delauney (FDC), Kevin Doherty (Hexagon), Andreas Niemann (Hexagon), Gwenaël Dunand (Intempora)	Main contributions and document consolidation
V1.0	28.02.2021	Oihana Otaegui (VIC), Marcos Nieto (VIC)	Document draft completed
F	28.02.2021		

Approval:

	Name	Date
Prepared	Oihana Otaegui (VIC), Marcos Nieto (VIC), Markus Bach (Valeo DE), Matthias Wellens (Valeo CTA)	28.02.2021
Reviewed	Esther Novo (VIC)	28.02.2021
Authorised	Oihana Otaegui (VIC)	28.02.2021

Circulation:

Recipient	Date of submission
GSA	01.03.2021
ACCURATE consortium	28.02.2021

Legal Disclaimer

CONTENTS

List of Figures

List of Tables

Executive Summary

The aim of ACCURATE is to develop a close-to-production high precision positioning on-board unit (OBU), which is based on tight heterogenous sensor fusion and can be integrated on automated driving platforms for any vehicle to reach SAE levels 4 and 5 of driving automation. The OBU will make use of the accuracy and integrity of the EGNSS components and services in a multifrequency approach, especially taking advantage of E5a and E5b. Additionally, a hybrid implementation of differential GNSS will be used as well as fusion with an IMU and perception sensors to enhance the capabilities of the positioning systems in adverse conditions. In a safety-critical approach, certification in accordance with the automotive industry functional-safety standard ISO 26262 will be considered during the design phase.

As part of the ACCURATE project, WP3 has the objective to assess the design options, and to select the solution architecture. The high-level architecture will be defined and broken down into low-level design details. The requirements established in D2.1 will serve as input for this WP.

Deliverable D3.1 shall define the preliminary OBU architecture, as a functional or logical reference architecture that includes SW and HW architecture designs. This reference will be used during the project as a basis and guidelines for development and testing. A second version of this work will produce deliverable D3.2 with the detailed OBU architecture, and the Design Justification File.

1 Introduction

1.1 Purpose of Document

As part of the ACCURATE project, WP3 has the objective of selecting the design options and design the architecture of the ACCURATE OBU.

This deliverable is the responsible of describing the technology employed by each-subcomponent. This high-level architecture will be used as a reference for the development and evaluation stages of the project, and used as the basis for the alpha prototype. This preliminary architecture will be refined in a second iteration, to be reported in D3.2.

1.2 Intended Audience

The dissemination level of D3.1 is public. Consequently, this deliverable is supposed to communicate the reference architecture that will guide the development of the ACCURATE OBU by forming a basis for the work of other WPs, especially WP4 and WP5.

1.3 Approach

This document contains contributions from all technical partners, which have gathered point of view, expertise in specific areas, in order to create a consensus view of the ACCURATE OBU architecture. The reference architecture has been debated and explored to identify potential miss-understandings in interfaces, expected functionalities, possible implementation options, and current status of developments and technology.

Then, HW and SW perspectives have been defined, where each component is defined according to its expected functionality. Low-level technical details have been left out of the scope of this document on purpose. Pointers to standards, spreadsheets or web resources are used to provide access to such details.

At the point of writing this deliverable, some implementation aspects could not be determined, and some debate is left for when the development stage start. In any case, this document contains the agreed reference architecture that all partners shall follow to meet the defined requirements.

2 Terminology

Glossary of acronyms

Acronym	Definition
3GPP	3rd Generation Partnership Project
AD	Autonomous Driving
ADAS	Advanced Driver Assistance Systems
ADASIS	ADAS Interface Specifications
ALKS	Automated Lane Keeping Systems
API	Application Programming Interface
DDBB	Data Base
DGPS	Differential GPS
DOF	Degrees of Freedom
ECU	Electronic Component Unit
EGNOS	European Geostationary Navigation Overlay Service
EGNSS	European GNSS Agency/Service
FUT	Function-Under-Test
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HIL	Hardware-in-the-Loop
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KPI	Key Performance Indicator
LIDAR	Light Detection and Ranging/Laser Imaging Detection and Ranging
LOC	Level of Confidence
ME	Measurement Engine
NTRIP	Networked Transport of RTCM via Internet Protocol
OBU	(Positioning) On-Board Unit
ODD	Operational Design Domain
PPP	Precise Point Positioning
RTCM	Radio Technical Commission for Maritime Services
RTMaps	Real-Time Multisensor Applications
SDK	Software Development Kit

SLAM	Simultaneous Localisation and Mapping
SUT	System Under Test
SWPE	(First level sensor fusion focusing on absolute positioning)
TCU	Telematics Control Unit
VUT	Vehicle Under Test
XiL	X-in-the-Loop (X stands for anything)

3 System architecture design

In this chapter, the high-level system architecture of the ACCURATE OBU is presented, which takes the functional requirements discussed in WP2 and defined in D2.1 into account. First of all, the technical approach of the ACCURATE project is explained, specifically introducing the two levels of sensor fusion. Afterwards, the high-level functional architecture of the OBU with the different inputs and outputs is described. Finally, details on the data flow and the utilized algorithms are given. The high-level architecture will be broken down into low-level design choices in the following chapters, which describe the details of the components and their employed technologies. An outlook at the actual implementation is provided by the last section of this chapter.

3.1 ACCURATE Approach

It is the main objective of the positioning OBU developed in the ACCURATE project to enable highly automated driving at SAE levels 4 and 5 by making use of EGNSS. On the path towards autonomous vehicles, precise and reliable absolute positioning is no longer a convenience but a critical requirement in order to stay inside the lane and avoid accidents. This shall be achieved under a variety of conditions described in the context of an extensive Operational Design Domain. Constrained environments with blocked or reflected GNSS signals, which are typical for urban scenarios, are particularly challenging for a GNSS-based positioning system. In order to overcome such limiting factors and satisfy the performance requirements regarding accuracy, integrity and availability, the ACCURATE OBU incorporates data from multiple sensors, which is fused on two separate levels.

On the first level of sensor fusion, raw GNSS signals are combined with GNSS correction data, measurements from an inertial sensor with six degrees of freedom and data from other sensors determining the vehicle state. The corresponding algorithm calculates the absolute position of the vehicle whose error grows with time and distance in environments with bad GNSS reception.

On the second level of sensor fusion, this position drift can be compensated by performing a localization relative to a map of the environment based on measurements from on-board perception sensors, such as LiDARs and cameras. The resulting output provides a position relationship between the ego vehicle and other road entities that is required for highly automated driving. Thus, the ACCURATE OBU can be integrated in automated driving platforms, e.g. Valeo’s Drive4U, and enable the subsequent steps of prediction, planning and generation of control and actuation orders.

More details on the two levels of sensor fusion as well as the involved components and modules are given in the following sections.

3.2 High-Level Functional Architecture

The high-level functional architecture of the ACCURATE OBU including the different inputs, the main internal components and the output is shown in Figure 1.

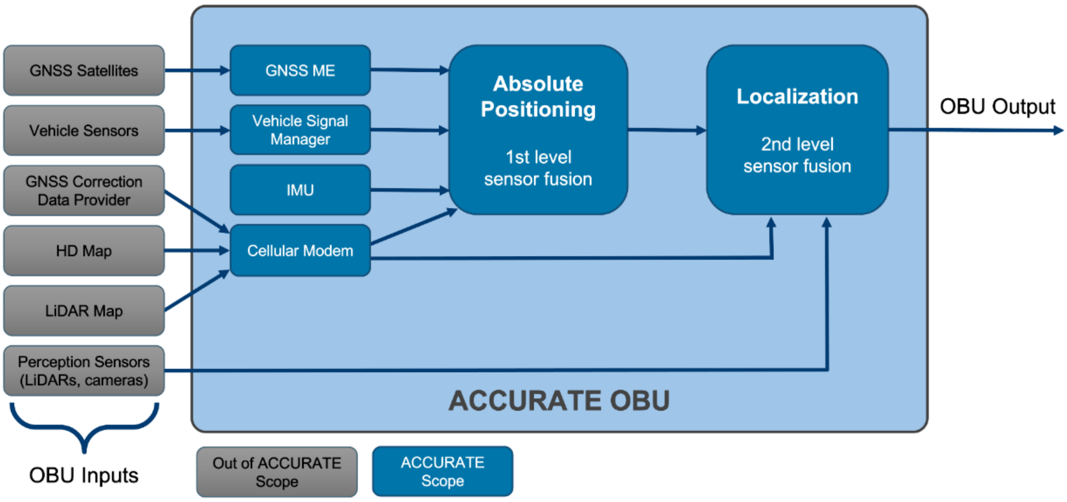


Figure 1: High-level functional architecture of the ACCURATE OBU.

The inputs of the OBU, which fulfil the corresponding requirements defined in D2.1, are the following:

- A GNSS ME (measurement engine) provides high-quality raw GNSS data from Galileo as well as GPS satellites and two different frequency bands with autonomous-driving precision and automotive safety compliance.
- A cellular modem receives accurate PPP (precise point positioning) correction data for both Galileo and GPS signals derived from a network of ground-base stations.
- Measurements of wheel ticks and steering angle are obtained from the corresponding vehicle sensors.
- An HD map (high-definition map) providing precise 3D data on the road features and the surrounding environment is downloaded by a cellular modem.
- A cellular modem receives a LiDAR map consisting of high-density point clouds of the road environment with accurate positions.
- Two kinds of data from perception sensors are used by the ACCURATE OBU: information on detected lanes extracted from camera images and accurate 3D point clouds of the environment generated by LiDARs with a large detection range and a wide field of view.

In addition to the described external inputs, the OBU integrates an IMU (inertial measurement unit) that consists of a triaxial accelerometer and a triaxial gyroscope measuring the acceleration and angular rate, respectively. The diverse data is transmitted to modules performing the two already introduced levels of sensor fusion, namely the absolute positioning and the localization.

In accordance with the requirements defined in D2.1, the OBU outputs the vehicle's position, velocity, heading and angular speed, as well as optionally its linear and angular acceleration. Additionally, the estimated position error and the protection levels of the position, which are used to evaluate the integrity of the system, are output.

3.3 Details on Data Flow and Algorithms

Figure 2 illustrates additional details of the OBU's architecture, specifically regarding the data flow and the main modules.

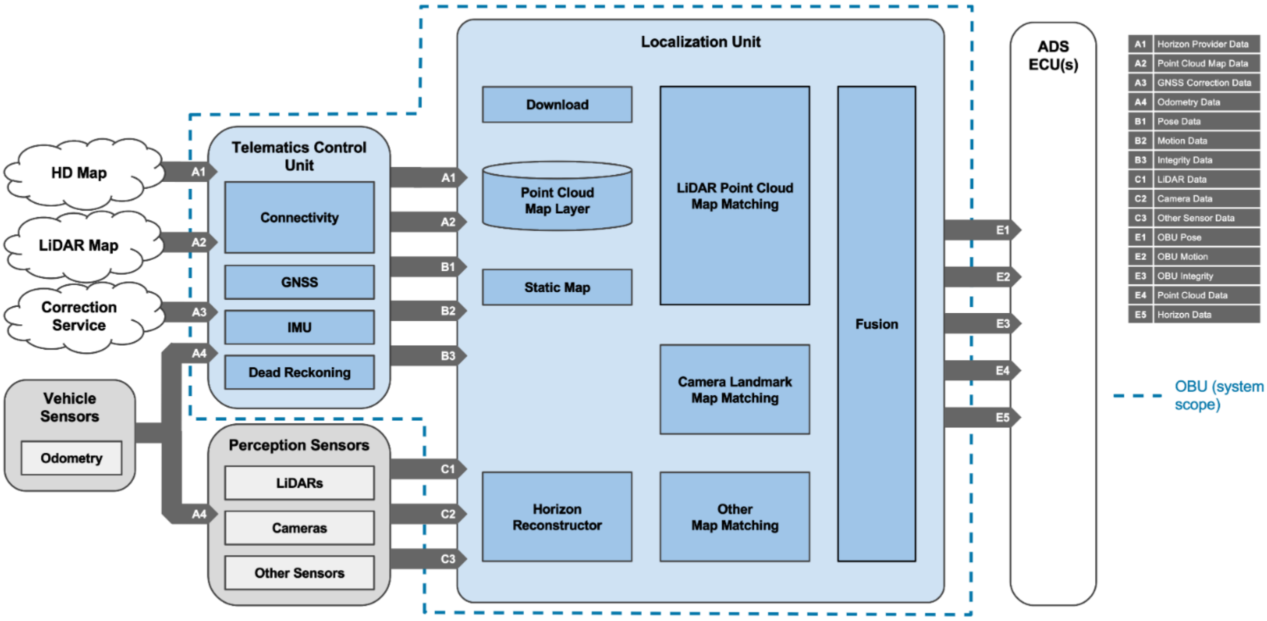


Figure 2: Data flow and main modules of the ACCURATE OBU.

Here, the **Telematics Control Unit** integrates the reception of raw GNSS signals, the connectivity provided by the cellular modem, the IMU and the first level of sensor fusion. While the HD and LiDAR map data is just forwarded to the Localization Unit, the inputs of GNSS correction data and odometry data from the vehicle sensors are fed into the absolute positioning algorithm. This algorithm consists of several parts. On the one hand, a PPP-based differential GNSS approach is applied to refine the raw, multi-frequency, multi-constellation GNSS measurements with the help of the downloaded correction data. On the other hand, the GNSS position is fused with the vehicle sensor and IMU measurements in a dead reckoning algorithm.

This enables a continuous positioning even in the case of lost or weak GNSS signals and a higher output rate of the position. Additionally, algorithms for GNSS integrity and authentication are included. As a result of the absolute positioning, the Telematics Control Unit outputs the pose data, comprising position, attitude and estimated position error, the motion data, comprising velocity, angular speed, linear and angular acceleration, as well as the integrity data, comprising protection levels of the position, to the Localization Unit.

While on the first level of sensor fusion, an absolute position in a global reference system is determined without any reference to the immediate vehicle environment, on the second level of sensor fusion, a localization relative to maps of the environment is performed. The latter is integrated into the **Localization Unit**. On the one hand, the static point cloud obtained from LiDAR sensors is compared to a downloaded LiDAR map in order to determine the accurate pose within the map. The focus is on this so-called LiDAR point cloud map matching. On the other hand, a camera landmark map matching is performed. Specifically, the information on detected lanes, which is extracted from camera images, is compared to corresponding features of a downloaded HD map in order to enable a localization in the map with lane-level accuracy. In principle, other map matching algorithms, which make use of different kinds of maps and features, could also be integrated to increase the precision and robustness of the positioning. However, this is currently not planned. The positions determined by the different algorithms are input into a fusion module that calculates a combined result of the overall sensor fusion, which is output to the units responsible for the automated driving. In addition to the pose and motion data, the OBU also outputs the integrity data, which is updated on the second level of sensor fusion beforehand, and potentially the static LiDAR point cloud as well as the e-horizon, which is obtained from a combination of the calculated position with cartography information.

The fusion of heterogeneous sensor types ensures an accurate and reliable positioning under a variety of conditions. On highways, the GNSS signals are rarely blocked and hence of great importance, along with the camera landmark map matching that helps staying inside the lane. In contrast to this, urban environments are typically challenging for GNSS and exhibit many structures, adding to the importance of LiDAR point cloud map matching. Since the ACCURATE OBU integrates all the sensor types relevant for mapping, its precise positioning might in the future even help resolving the challenge of updating the utilized maps via upload of information to the cloud.

3.4 Implementation Decisions

The following list provides an overview of the concrete decisions regarding the implementation of the ACCURATE OBU, which will be further described in the following chapters:

- Valeo’s Vulcano-5G **Telematics Control Unit** integrates a 5G **cellular modem**, which provides the necessary **connectivity**, and an embedded application processor. On this processor, Hexagon’s software positioning engine, which performs the **absolute positioning**, and the **vehicle signal manager** will be run.
- FDC’s daughterboard will be mounted on the TCU and host an ASM330LHH **IMU** from STMicroelectronics and an STA8100GA **GNSS receiver**, which belongs to STMicroelectronics’ Teseo V family.
- Hexagon’s TerraStar X **GNSS correction service** will be used.
- As **perception sensors**, Valeo will provide SCALA laser scanners and cameras.
- The **Localization Unit**, on which the second level of sensor fusion will be performed, consists of an embedded PC and an EB robinos device, which enables the reception of **HD map** data. RTMaps from Intempora will be used as middleware with modules for the LiDAR- and camera-based matching algorithms developed by Valeo.
- The ACCURATE OBU will be integrated in Valeo’s Drive4U automated driving platform for testing and demonstration.

4 Hardware architecture design

This section will introduce the high-level view on the ACCURATE OBU hardware architecture and will also discuss the implementation foreseen during the first year of the project.

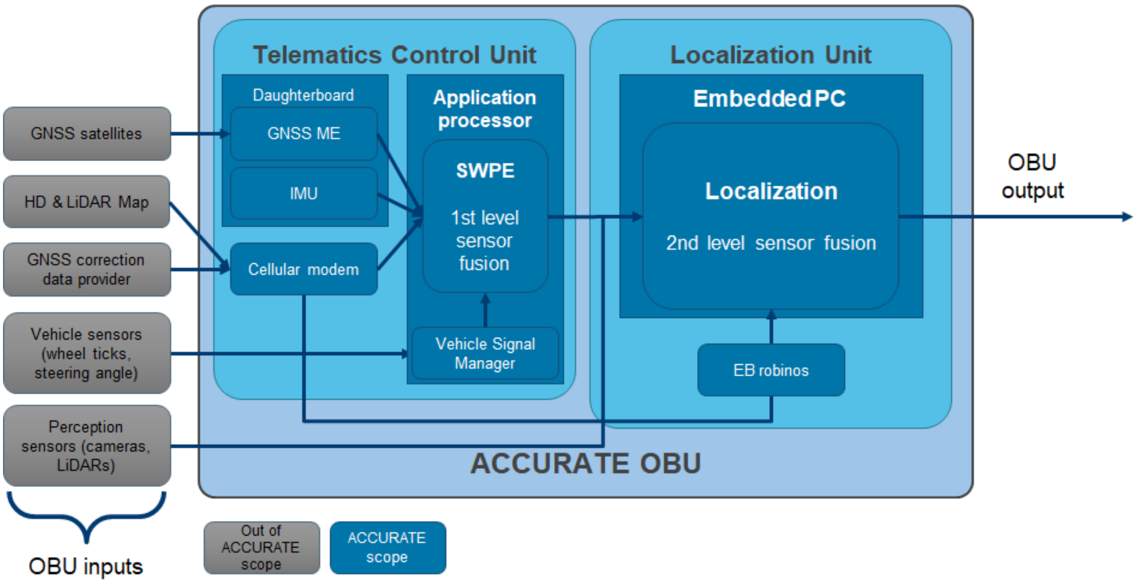


Figure 3: Implementation view on ACCURATE OBU architecture.

Figure 3 introduces the implementation view of the system architecture discussed in chapter 3.

The inputs of the OBU are the same as introduced in the previous chapter. Same goes for the outputs to the applications. However, the internal architecture of the OBU shown in Figure 3 has been adapted to the implementation as foreseen for the ACCURATE project.

The high-level architecture with two major steps of sensor fusion is also present in the implementation view. The TCU with its subcomponents and a daughterboard hosting the GNSS measurement engine (ME) and the IMU (Inertial Measurement Unit) will be in charge of input data preparation and the first level of sensor fusion. The Localization Unit will apply the second level sensor fusion considering perception sensors and the HD as well as LiDAR map.

The following subsections will introduce each main component of the system in more detail. The order will go from left to right when looking at Figure 3 or, in other words, from RF and antennas towards the processing and the applications, e.g., the automated driving ECUs, retrieving the OBU output. For each subcomponent we will discuss two main aspects: The component's function and details on its foreseen implementation in the ACCURATE project.

Considering the focus use case of automated driving L4/L5 for the ACCURATE project, the topic of functional safety is important. The ACCURATE project will take a diversified approach here. The concepts that the project develops shall be able to meet functional safety requirements while the prototyping implementation may not always do so. The latter is driven by the need to focus the project resources but also by simple component availability. Often, functional safety capable components are developed by suppliers in a second step while the basic functionality and its improvement over the state-of-the-art can be demonstrated and studied with earlier samples. Thus, the more detailed evaluation of how the architecture and, in particular, the implementation needs to be extended to meet functional safety requirements will be left for the second year of the project.

Additionally, a comprehensive safety case evaluation as per ISO26262 will need to consider the complete system including the use case implementation. Since the use case is not fully within the ACCURATE scope the project will also not execute a complete safety case evaluation but will focus on concepts that will enable an ASIL rating for the ACCURATE OBU.

Another direct consequence of the high precision requirements of ACCURATE's main target use case automated driving is the importance of measurement sample timestamps. The automated vehicle will already have moved for some distance in the time between measuring a new GNSS or IMU data sample and the time at which it will really be available to the fusion algorithms. Thus, the end2end system needs to be able to determine accurate timestamps for each measurement sample and to consider these throughout the complete processing chain. The fact that the involved components' local clocks may drift apart or may be based on different reference times, e.g., considering, or not considering leap seconds adds further complexity. The ACCURATE consortium is aware of the relevance of this topic and will properly address it in its prototyping activities. However, due to the early stage of the implementation work we do not comment on it here.

4.1 Antenna

Traditional automotive grade GNSS antennas are not designed for high precision positioning solutions and do not provide the necessary characteristics to allow for consistent sub-metre level accuracy with regards to signal support, phase centre characteristics and gain pattern. Existing survey grade antennas, although suitable from a performance standpoint, are not built to automotive standards or available in a form factor that allows for integration into a production vehicle solution. Hexagon has developed an automotive grade, multi-frequency, multi-constellation GNSS antenna that is consistent with requirements for high precision and will be utilized in the ACCURATE project.

Specifications of the GNSS-1500 samples for integration with the OBUs:

- 5M RF cable with a male SMA connector

- Input voltage range of +4.5 to +6.5 VDC with a current draw of ~50 mA (typical)
- Pass Bands:
    - 1545 – 1610 MHz
    - 1166 – 1254 MHz
- Signals Supported:
    - GPS L1/L2/L5
    - Galileo E1/E5a/E5b
    - GLONASS L1/L2
    - BeiDou B1I/B1C/B2I/B2a/B2b
- Physical Dimensions:
    - Element: 43 x 43 x 12.5 mm
    - Element and LNA: 61 mm D x 15 mm H
    - Units will be provided in enclosure

4.2 GNSS Receiver

For the ACCURATE Project, FDC is in charge to develop a GNSS receiver. This receiver will be implemented on a daughter board that will be mounted on the Valeo Vulcano-5G prototyping unit.

4.2.1 Functional View

In the framework of the ACCURATE project, the main functions of the GNSS receiver are:

- Reception and processing of GNSS signal (multi-band and multi-constellation),
- GNSS Raw measurements and navigation message output on UART in RTCM10403 format (with additional proprietary messages),
- Processing of Open Service Navigation Message Authentication (OS-NMA),
- Implementation of additional anti-spoofing and jamming detection techniques,
- Providing a PPS signal.

Reception of GNSS signal: The GNSS receiver is compatible with all different GNSS constellations. It supports one among these 5 following configurations.

Table 1: Frequency plan.

	L1				L2				L5		
	GPS	GLO	BEID	GAL	GPS	GLO	BEID	GAL	GPS	BEID	GAL
	L1	G1	B1I	E1	L2C	G2	B2I	E5b	L5	B2a	E5a
Configuration 1	**X**	**X**	**X**	**X**	**X**	**X**	**X**				
Configuration 2	**X**	**X**	**X**	**X**	**X**		**X**				
Configuration 3	**X**	**X**	**X**	**X**					**X**	**X**	**X**
Configuration 4	**X**	**X**	**X**	**X**	**X**			**X**			
Configuration 5	**X**	**X**	**X**	**X**	**X**		**X**	**X**			

RTCM messages: The RTCM messages will be provided through a UART and will be conform with the RTCM 10403 protocol with additional proprietary messages.

Open Service Navigation Message Authentication: The Galileo OS-NMA will be processed according to the SIS ICD V2.1. The receiver will include an interface allowing to fill an OS-NMA public key through the UART interface. The OS-NMA will run on the receiver configured to use Galileo frequencies.

Anti-spoofing/Jamming detection: Additional anti-spoofing will complement the OS-NMA processing block: PVT cross check (constellation agility), C/N0 monitoring, Time jump monitoring and TRAIM, dynamic consistency. These anti-spoofing techniques will monitor several parameters taking into account the receiver location and the environment characteristics in order to detect additional spoofing attacks on GNSS not covered by OS-NMA and reinforce OS-NMA for some types of attacks.

Jamming detection is performed through AGC monitoring and spectral analysis.

A global level of confidence (LOC) indicator is provided in the output. This LOC is a score between 0 to 100 and is the result of the fusion of all anti-spoofing measures. A LOC score under 60 means that the GNSS signals are compromised. In this case an ALARM digital output is activated.

PPS signal: A PPS output will be generated every second and synchronized with UTC time.

4.2.2 Module Implementation

The FDC GNSS module architecture is built around the STA8100GA GNSS receiver and the STA5635A radio frequency front end. The STA8100GA is a new generation GNSS receiver, developed by STMicroelectronics and it belongs to the Teseo V family.

The figure below shows the interfaces between the main components that compose the FDC module.

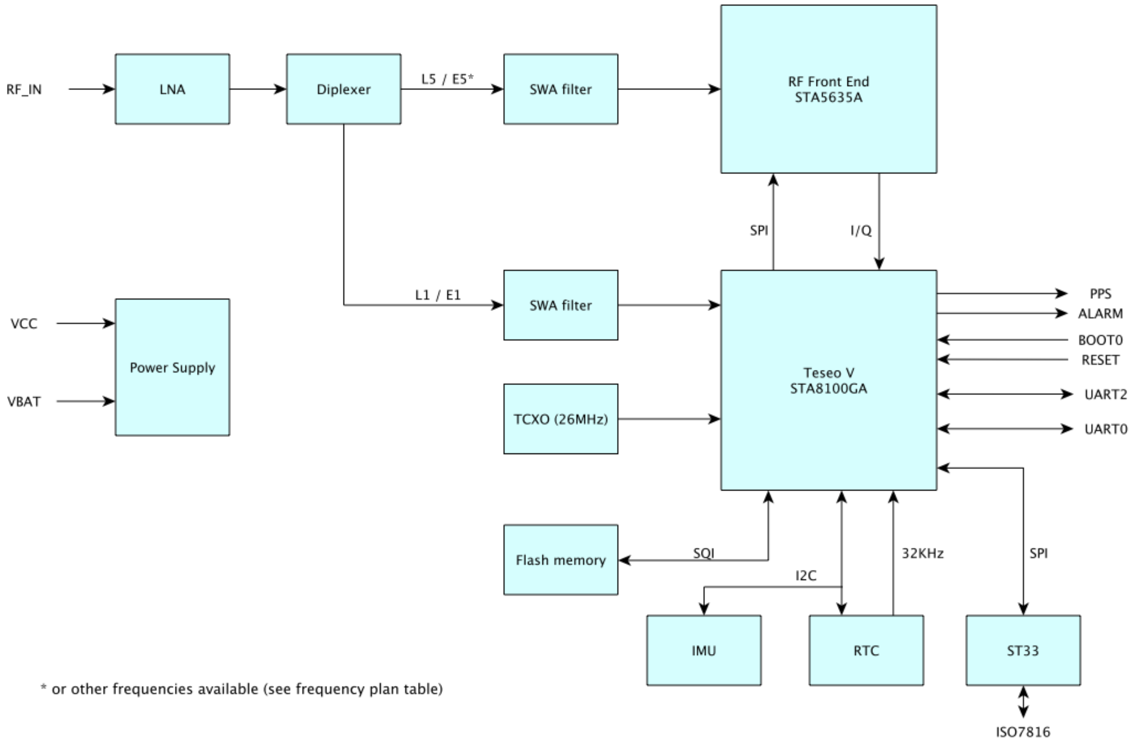


Figure 4: Teseo V module architecture.

The GNSS signal is amplified by an LNA (Low Noise Amplifier). The LNA output is shared by the diplexer in two signals. One in the L1/E1 band to feed the Teseo V and the other in the L5/E5 band to feed the STA5635A. It is possible to choose another band than L5/E5 in accordance with the frequency plan.

The STA8100GA firmware and the configuration data are stored in the flash memory.

The RTC provides a 32KHz that feeds the Teseo V XTAL input. This RTC also provides, through an I2C bus, an independent GNSS time source used for OS-NMA at startup.

The ST33, a secure MCU, is used as a secure NVM.

4.2.3 Daughter board implementation

The daughter board accommodates the FDC GNSS receiver and these additional functions:

- Antenna power,
- Firmware update,
- Power supply,
- RF signal connection,
- Vulcano motherboard connection,
- IMU.

Figure 5 shows the architecture of the daughter board.

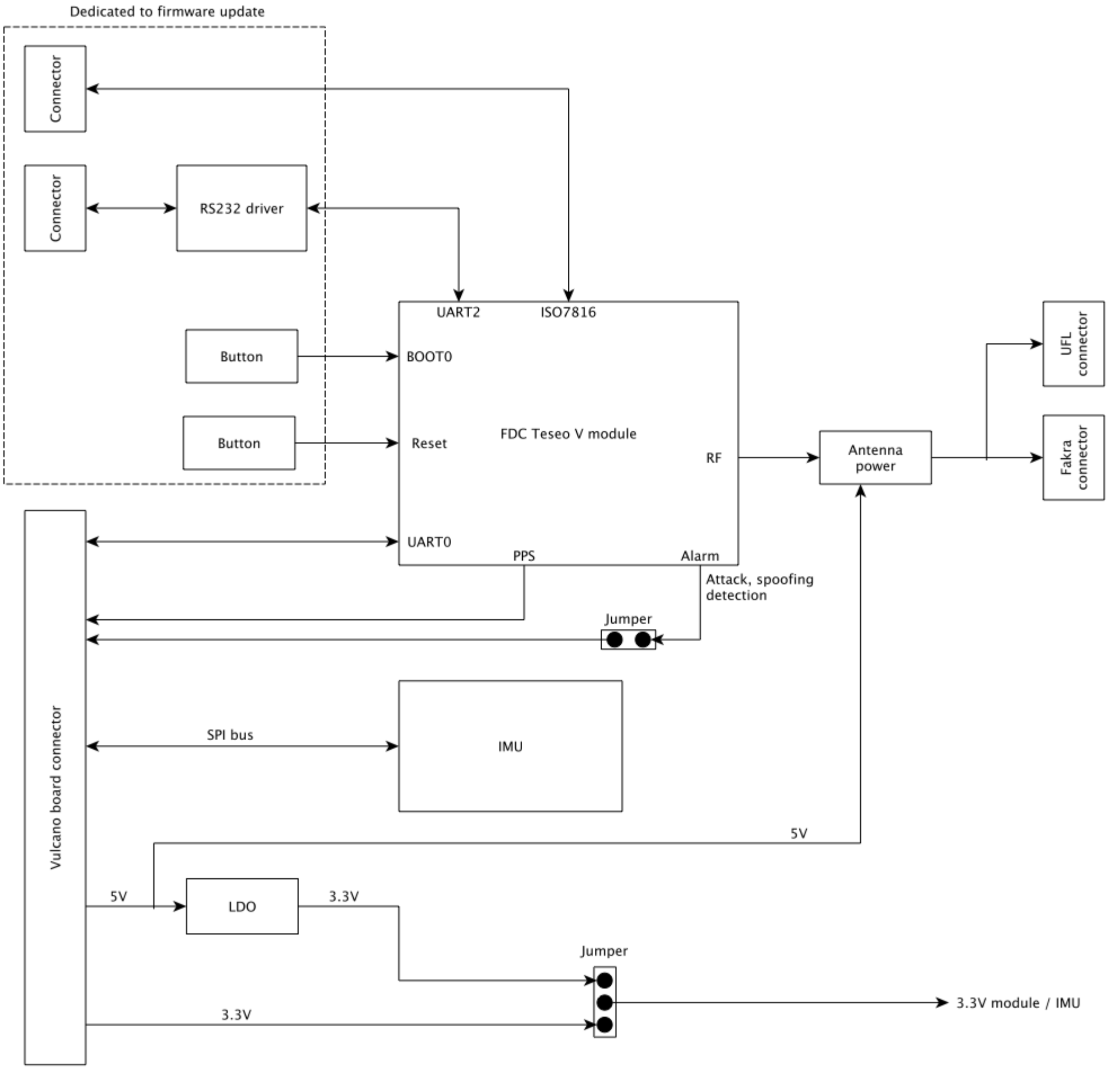


Figure 5: Daughter board architecture.

The FDC GNSS module provides a digital output ALARM which indicates if an attack is detected. A jumper will output this information if necessary. Another jumper will allow to choose the main power supply (5V or 3.3V).

4.3 IMU

4.3.1 Functional View

The IMU is to provide inertial measurement data to the positioning engine. The inertial measurement data shall be in the form of three accelerometer measurements and three gyroscope measurements, together composing a 6 degree of freedom (DOF) IMU, which can measure 3-dimensional acceleration and rotation. The measurements should be provided to the position engine at a rate of approximately 100 Hz, and shall be accompanied by a timestamp with a high degree of accuracy relative to GPS time (<1 ms error). Additional context such as IMU status and temperature may also be useful.

The above specifications can be met by mounting a 6 DOF IMU on a rigid part of the vehicle body, with a fixed spatial offset from the GNSS antenna. To achieve accurate timestamping, most IMUs use one of the following techniques:

a. Provide a 'Data Ready' pulse before each set of IMU data. The Data Ready pulse can be read by a CPU that also maintains accurate GPS time which would provide the timestamp
b. The IMU accepts a PPS (or another known rate) input pulse which it uses to keep accurate time with, and provides the timestamped data directly
c. The IMU accepts a 'Collect Data' pulse that can be sent at a known GPS time, and the IMU will capture data accurate to the time the pulse is received

4.3.2 Implementation View

The IMU will be implemented on the daughter board. Its SPI Bus will be directly output on the Vulcano connector. This allows the application processor assembled to the Vulcano board to read the IMU data without the limitation dues to the FDC GNSS module (especially the IMU sampling rate).

It is foreseen to use an ASM330LHH from STMicroelectronics for the first stage of ACCURATE project.

4.4 Correction Data Provider

The GNSS correction service used for this project is Hexagon's TerraStar X. It combines existing TerraStar (PPP) global clock and orbit data with regional ionospheric correction data from Hexagon's vast network of HxGN SmartNet reference stations to provide fast convergence to lane-level accuracy with low-cost GNSS receivers. TerraStar X also enables integrity and authentication for safety-critical applications like autonomous driving and aviation.

Correction data for GPS and Galileo are supported. Corrections for additional constellations may be available during the ACCURATE project at a later stage.

The correction data is hosted on a Hexagon server and access is provided by NTRIP protocol.

4.5 Telematics Control Unit

The Telematics Control Unit (TCU) is an important part of the ACCURATE OBU since it enables access to updates for GNSS correction data, the static LiDAR point clouds, and the HD map.

On the implementation side, the TCU will also host additional functionality such as the GNSS measurement engine.

4.5.1 Functional View

While a TCU may host plenty of additional features on the implementation side, we will focus here on its communication capabilities, which are still its core function.

The TCU connects the ACCURATE OBU with remote communication nodes to exchange data. Most notably the TCU will transfer GNSS correction data and LiDAR and HD map updates. Further extensions may enable the system to provide new observations back to the map servers to reflect local changes and to share map updates with the larger community. Additionally, general communication services such as mobile Internet access will enable further improvements to almost all use cases of the ACCURATE OBU since mobility is an inherent characteristic.

The TCU will also support wired or wireless local area networks to connect with other nodes in the vehicle that, e.g., host the HD map and process its updates.

Finally, the TCU usually hosts a processor. It will at minimum augment the communication services, e.g, with cybersecurity features such as a firewalling but may also run processing such as the first level sensor fusion depending on the implementation.

4.5.2 Implementation View

The ACCURATE project will use Valeo’s Vulcano-5G TCU.

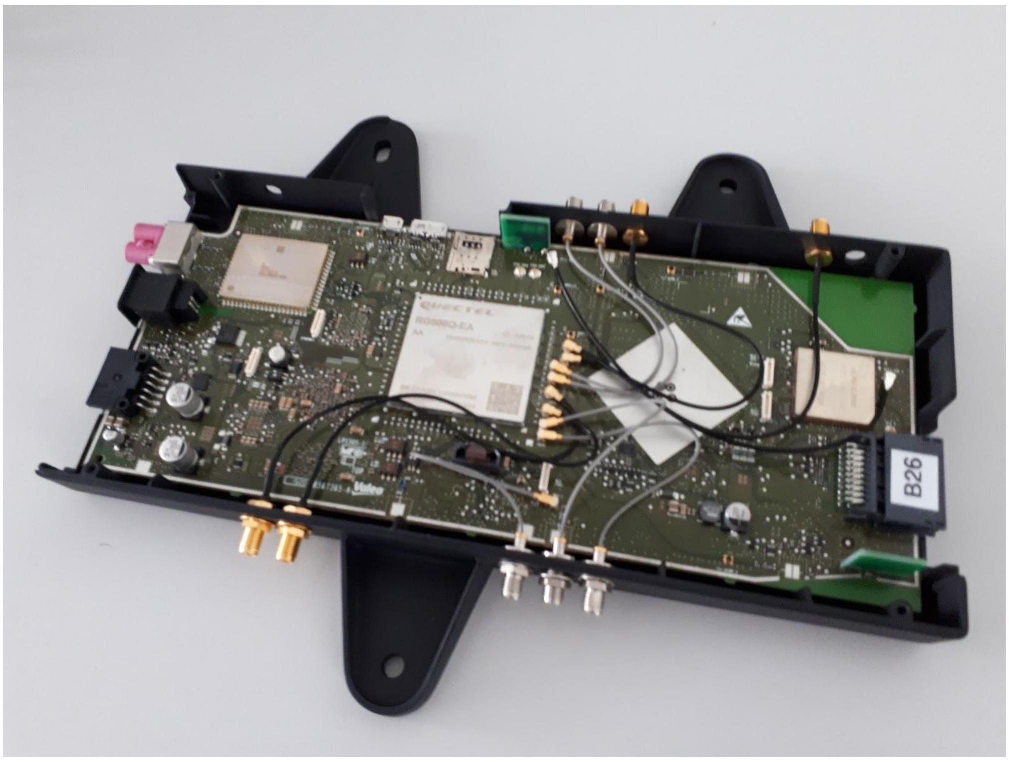


Figure 6: Vulcano-5G TCU board.

Figure 6 shows Valeo’s Vulcano-5G, which is a 5G-capable TCU that is being developed to be a flexible basis for prototyping and advanced engineering work.

The Vulcano-5G TCU variant that will be used for the ACCURATE-project provides 5G cellular mobile radio, an embedded application processor, automotive ETH and a daughterboard connector for further extensions. It offers several further features that we skip here since they will not be used during the ACCURATE project.

Vulcano-5G’s modem is a 3GPP Rel. 15-compliant device that supports multi-mode cellular communication based on 5G, but is also backwards compatible to 2G, 3G, and 4G. It will support dedicated services such as the download of GNSS correction data or HD map updates but will also provide standard Internet access to the remaining ACCURATE OBU and further nodes in the vehicle network.

On the vehicle network side, Vulcano-5G supports among other interfaces automotive Ethernet to provide high-speed connectivity to the localization unit and the remaining vehicle. The system supports Gigabit-speeds based on the IEEE 1000BaseT1 standard but is also backwards compatible with IEEE 100BaseT1. Connectivity to standard non-automotive Ethernet (IEEE 1000BaseT or IEEE 100BaseT) can easily be enabled using a media converter bridging both standards.

If certain software components are needed, e.g., to manage the download of GNSS correction data, these components will run on Vulcano-5G’s application processor. The same component will also run the SWPE implementing the first level sensor fusion focusing on absolute positioning. To manage the retrieval of vehicle data such as wheel ticks and the steering angle and to provide it to the SWPE, the application processor will also run the Vehicle Signal Manager, an additional software component.

As part of the ACCURATE project, the Vulcano-5G platform will be further improved in order to ensure smooth interoperability with the daughterboard hosting the GNSS ME and the IMU as introduced above. Vulcano-5G will offer a daughterboard connector to interface with the dedicated daughterboard and to connect the GNSS ME and the IMU to the application processor. Additionally, general Vulcano-5G platform maintenance is carried out to integrate updated component hardware releases of the main components (e.g., the latest release of the automotive ETH 1000BaseT1-PHY).

4.6 Perception Sensors

The perception sensors are themselves not a part of the ACCURATE OBU but their data is required for the localization relative to given maps on the second level of sensor fusion. A suitable placement and quantity of sensors will be investigated in the later stages of the project.

On the one hand, a dense 3D point cloud of the vehicle environment has to be provided by LiDAR sensors. Specifically, Valeo’s SCALA LiDAR will be used, which is the first laser scanner for the automotive volume production and thus supports the ACCURATE objective of developing a real product to be deployed in any vehicle. It is a key enabler for automated driving applications and works under a variety of conditions (day and night, low and high speed). The second generation of SCALA has the following specifications:

- Wavelength: 905 nm
- Detection range: 150 m (reflective targets)
- Horizontal field of view: 133°
- Vertical field of view: 10°
- Scan rate: 25 Hz
- Number of layers: 16
- Distance resolution: < 100 mm
- Interface: Ethernet

On the other hand, the ACCURATE OBU requires input from a camera that is capable of detecting lanes in order to perform lane-level localization relative to an HD map. Details on the foreseen implementation cannot be given for confidentiality reasons.

4.7 Localization Unit

On the one hand, the localization unit has to include a computing device, which can run the middleware RTMaps and the algorithms that perform the second level of sensor fusion. During the first phase of the project, an embedded PC will be used for this purpose. After the whole computing pipeline is developed and tested, the possibility of using an embedded computing board will be investigated.

On the other hand, the localization unit has to include an EB robinos hardware device, which acts as a client to receive the horizon data from servers of different HD map providers.

4.8 ACCURATE-OBU implementation

The currently foreseen prototyping implementation of the ACCURATE OBU will cover its full functionality. It will leverage automotive grade embedded components where available with the required performance but will also benefit from the flexibility of PC-based prototyping for those parts where algorithm development is the main focus.

All components have been chosen to enable efficient integration for the foreseen software architecture as introduced in the next section but also to allow for powerful XiL verification work as described in section 6.

5 Software architecture design

The software architecture of the ACCURATE OBU is visualized in Figure 7.

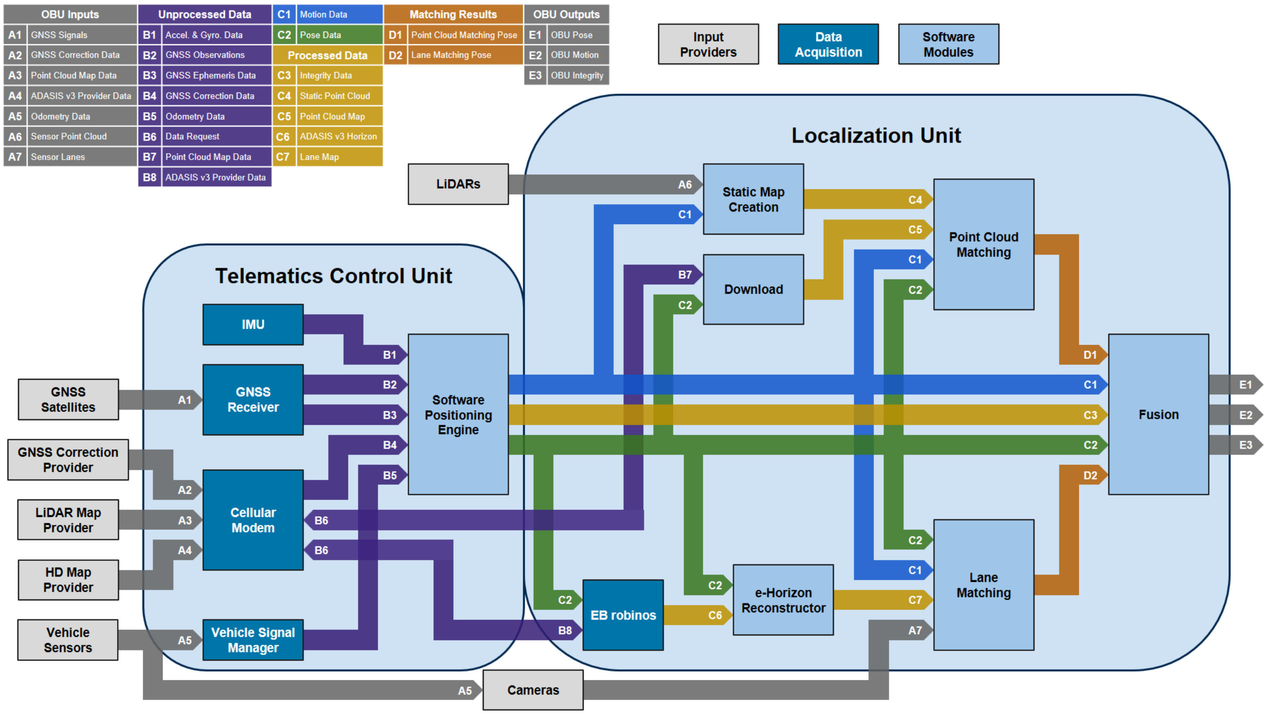


Figure 7: Software architecture of the ACCURATE OBU.

Next subsections shows the features or RTMaps, the selected middleware to run the proposed SW architecture.

5.1 Conceptual View (RTMaps)

5.1.1 Middleware role

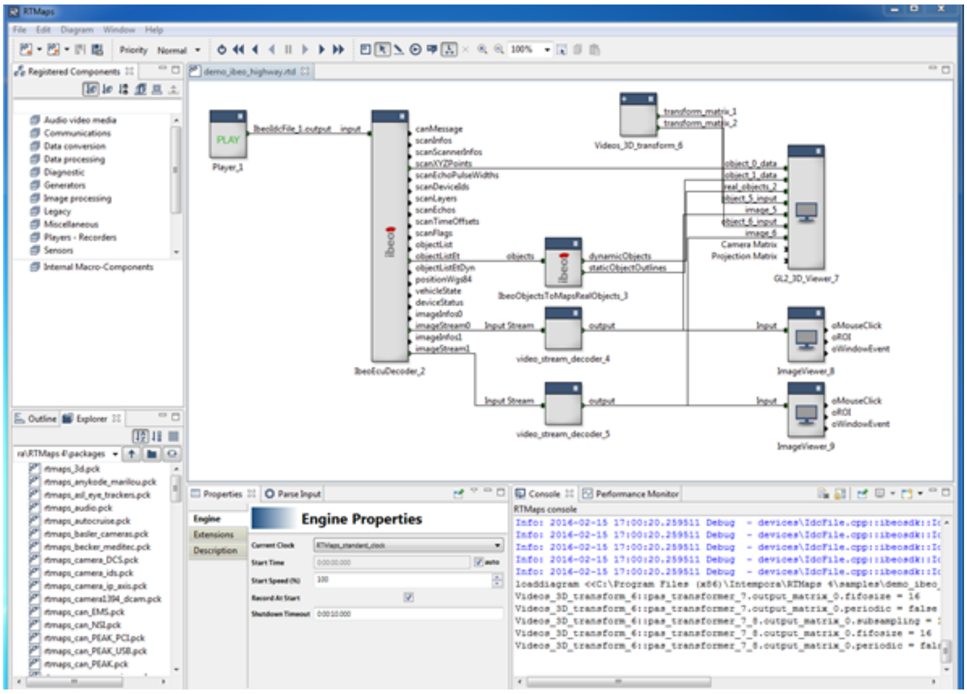
The number of sensors used for ADAS applications has increased in the last few years. Now applications use radars, lidars, GPS, high definition stereo cameras, lasers, IMU, CAN Bus, eye trackers, V2V and V2I communication, etc. The problem is how to read all of them within the same application and especially how to synchronize them despite their very different nature (Figure 7).

To achieve reading from multi-modal sensors, RTMaps middleware is fully asynchronous – each component runs in its own thread – so that any component can react to any data stream, whatever sampling rate it may have. This is the only way to follow the natural pace of each data. This design uses internally blocking calls, removing any extra latency that could happen when using polling methods. RTMaps middleware also defines reading policies to synchronize data streams. While the default policy – reactive – works perfectly fine in most case, the user can use one of those:

- **Reactive reading**: a component with multiple inputs will read every time a new data sample is made available on any one of its inputs.
- **Synchronized reading**: a component with multiple inputs will process one sample from each input when data sample with the same timestamps (plus or minus some configurable tolerance) are available on its inputs. This behaviour is made for data fusion and allows re-synchronization of the data streams at any point downstream in the diagram, whatever the latency of the various upstream data channels
- **Triggered reading**: a component with multiple inputs will read when a new data sample is made available on a given input. It will then resample the data on its other inputs through non-blocking reading

5.1.2 Component-based GUI

RTMaps will be used in the ACCURATE project as a common platform for data acquisition, record, storage and replay. Once the application has been created with the graphical RTMaps studio and all the already available off-the-shelf components, the Runtime execution engine can be deployed almost everywhere, from standard computer to embedded devices.



Engine Properties

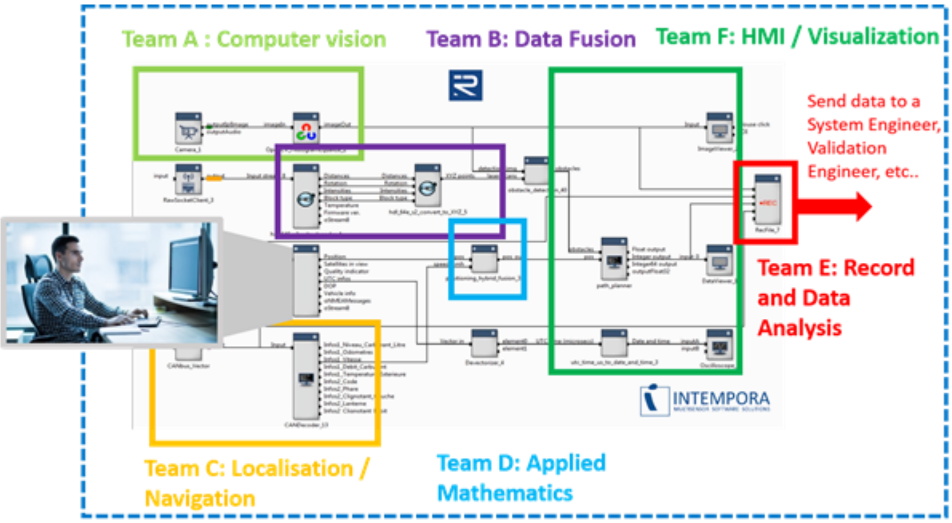
Current Clock: RTMaps_standard_clock
Start Time: 0:00:00.000 ☑ auto
Start Speed (%): 100
Record At Start: ☑
Shutdown Timeout: 0:00:10.000

```
Info: 2016-02-15 17:00:20.259511 Debug  - devices\IdcFile.cpp::ibeosdk::Ic
Info: 2016-02-15 17:00:20.259511 Debug  - devices\IdcFile.cpp::ibeosdk::Ic
Info: 2016-02-15 17:00:20.259511 Debug  - devices\IdcFile.cpp::ibeosdk::Ic
Info: 2016-02-15 17:00:20.259511 Debug  - devices\IdcFile.cpp::ibeosdk::Ic
Info: 2016-02-15 17:00:20.259511 Debug  - devices\IdcFile.cpp::ibeosdk::Ic
loaddiagram <<C:\Program Files (x86)\Intempora\RTMaps 4\samples\demo_ibeo_
Videos_3D_transform_6::pas_transformer_7.output_matrix_0.fifosize = 16
Videos_3D_transform_6::pas_transformer_7.output_matrix_0.periodic = false
Videos_3D_transform_6::pas_transformer_7_8.output_matrix_0.subsampling = 
Videos_3D_transform_6::pas_transformer_7_8.output_matrix_0.fifosize = 16
Videos_3D_transform_6::pas_transformer_7_8.output_matrix_0.periodic = fal
```

Figure 8: RTMaps Studio.

The advantage of using a graphical user interface is twofold. Firstly, it allows the user to quickly construct an application by using drag and drop techniques and wiring components to each other. Realising a simple demonstration with a camera and an IMU only takes a few minutes whereas using only hand-written code with dedicated libraries would take weeks.

Secondly, it allows the team to focus on interfaces. This is a very important point since it defines boundaries and clarifies the work between teams. In big projects like the ACCURATE project, strict definitions about interfaces are necessary due to the number of partners. The interface for components is composed of inputs, outputs and properties. Once the interface of a task is defined, changing an algorithm for another is easy as one component can be replaced by another.


Team A : Computer vision
Team B: Data Fusion
Team F: HMI / Visualization
Send data to a System Engineer, Validation Engineer, etc..
Team E: Record and Data Analysis
INTEMPORA
Team C: Localisation / Navigation
Team D: Applied Mathematics

Copyright INTEMPORA S.A.

Figure 9: Collaborative development using RTMaps.

5.1.3 Collaborative development

Extending the component library is done through the **SDK**, whose purpose is to expand the capabilities of the middleware by the creation of new components. In RTMaps for example, the SDK is available for both C++ and Python. Thanks to this SDK, the user can integrate his own code into a component and use it directly in this diagram.

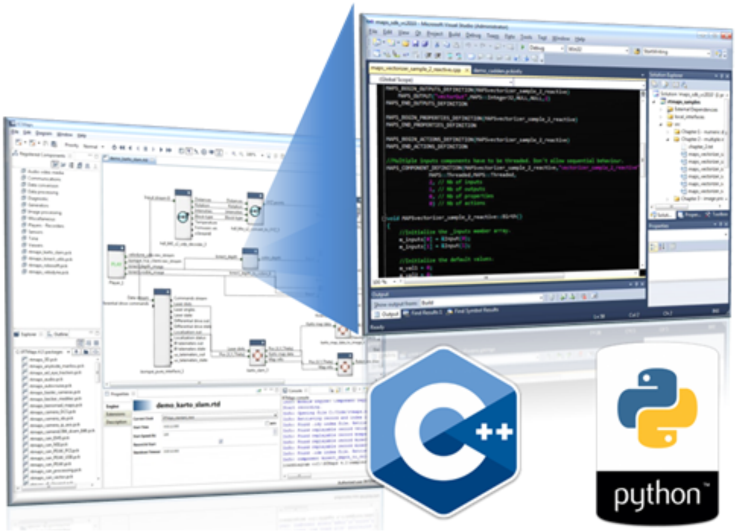


Figure 10: RTMaps multi-platform SDK.

Once a new component has been created, it can be shared with others. When using C++, each component is compiled code, which means that only the binary code is used in the middleware and so the IP is preserved. Anybody can share his work while keeping the source secret.

5.1.4 Synchronization

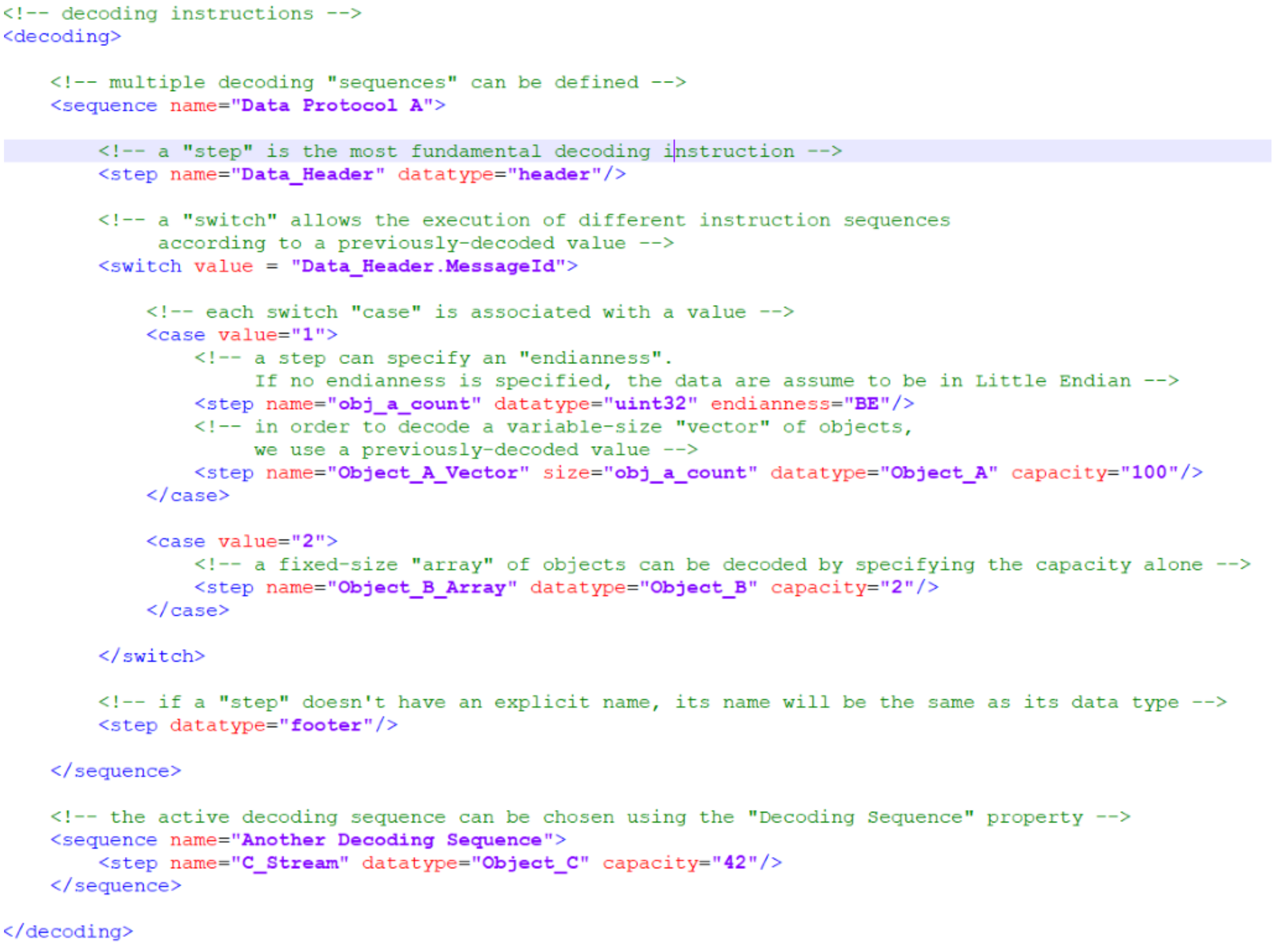
Using a middleware allows to be very accurate about the timing of your data. For example, RTMaps affects two timestamps to the data: the timestamp and the time of issue.

- The timestamp is the intrinsic date of the sample. It is as close as possible to the date of occurrence of the real data which the sample corresponds to. It is often supplied by the first component that created the sample (i.e. the acquisition component). The timestamp remains unmodified while the sample goes through the different components of the processing chain. The timestamp often corresponds to the date where the data is available in system memory.
- The time of issue is the date corresponding to the last time the sample was output from a component. Therefore, this date increases as long as the sample runs through the different processing components.

Knowing with precision the time and date of your data is essential to perform synchronized readings (see previous section), but it is also useful to estimate the latency of your data or know the processing time of a component which is really vital in real-time applications.

5.1.5 Network interfacing

RTMaps has a custom ethernet decoder that allow the partners to decode a custom stream through a socket. This decoder uses XML as struct definition and runtime decoding. This functionality will be useful for the ACCURATE project to receive external sources that do not run RTMaps



```
<!-- decoding instructions -->
<decoding>

    <!-- multiple decoding "sequences" can be defined -->
    <sequence name="Data Protocol A">

        <!-- a "step" is the most fundamental decoding instruction -->
        <step name="Data_Header" datatype="header"/>

        <!-- a "switch" allows the execution of different instruction sequences
             according to a previously-decoded value -->
        <switch value = "Data_Header.MessageId">

            <!-- each switch "case" is associated with a value -->
            <case value="1">
                <!-- a step can specify an "endianness".
                     If no endianness is specified, the data are assume to be in Little Endian -->
                <step name="obj_a_count" datatype="uint32" endianness="BE"/>
                <!-- in order to decode a variable-size "vector" of objects,
                     we use a previously-decoded value -->
                <step name="Object_A_Vector" size="obj_a_count" datatype="Object_A" capacity="100"/>
            </case>

            <case value="2">
                <!-- a fixed-size "array" of objects can be decoded by specifying the capacity alone -->
                <step name="Object_B_Array" datatype="Object_B" capacity="2"/>
            </case>

        </switch>

        <!-- if a "step" doesn't have an explicit name, its name will be the same as its data type -->
        <step datatype="footer"/>

    </sequence>

    <!-- the active decoding sequence can be chosen using the "Decoding Sequence" property -->
    <sequence name="Another Decoding Sequence">
        <step name="C_Stream" datatype="Object_C" capacity="42"/>
    </sequence>

</decoding>
```

Figure 11: RTMaps XML struct definition runtime decoding network interfaces.

5.2 Absolute Positioning

The positioning engine, which performs the first level of sensor fusion, is delivered by Hexagon as a shared library (.so) file. This library is then linked in to the software running on the application processor and provides position computation functions for that application. The hosting application is responsible for routing the inputs to the library via the engine API as well as making use of the computed position solution. The engine requires the following inputs, which can also be seen in Figure 7 above:

1. **Deployment Configuration File** – This is provided to the engine during once initialization and contains configuration information specific to the deployment (eg. IMU-to-Antenna lever arms).

2. **GNSS Observations** – Measurements output by the GNSS receiver must be read and provided by the host application to the positioning engine library in the specified format using the appropriate API function.
3. **GNSS Ephemeris** – Ephemeris data output by the GNSS receiver must be provided to the positioning engine library.
4. **GNSS Correction Data** – Correction data received from the correction service must be provided to the positioning engine in order to compute a corrected solution.
5. **IMU Data** – IMU measurements must be accurately time-stamped and provided to the positioning engine in the format specified in the API.
6. **Vehicle Sensor Data** – Other vehicle sensor data (eg. Wheel ticks, steering angle) may optionally be provided to the library in the format specified in the API. Doing so will improve the accuracy of the solution during GNSS outages.

In response to each set of observation data (GNSS observations and IMU observations), the positioning engine library will compute a position solution in the format specified in the API. The solution includes the computed protection level for that solution. The GNSS solution reports the position of the GNSS antenna phase centre. The INS solution reports the position of the IMU centre of navigation.

5.3 Localization

The RTMaps structure of the second level of sensor fusion, which will be run in the Localization Unit, is visualized in Figure 7 above. Here, the term **pose** denotes a combination of global position (longitude, latitude, altitude), attitude and the corresponding standard deviations. The term **motion** denotes a combination of velocity, angular speed, linear and angular acceleration as well as the corresponding standard deviations. The **integrity** data comprises protection levels of the position and possibly also those of other quantities. As a result of the absolute positioning, the Telematics Control Unit outputs pose, motion and integrity data to the Localization Unit. This data is used by different software modules that are explained in the following paragraphs.

The **Static Map Creation** module uses the motion data to classify the 3D points from the point cloud provided by LiDAR sensors regarding their movement. This enables filtering out dynamic points thus creating a static point cloud.

The **Download** module uses the current position included in the pose data to create a wish list of needed point cloud map tiles. A corresponding request is sent to the LiDAR map provider via the connectivity provided by the cellular modem such that the required point cloud map data can be received and decoded into the point cloud map.

The **Point Cloud Matching** module uses the motion data and compares the static point cloud provided by the Static Map Creation module with the point cloud map provided by the Download module to achieve a localization relative to the map. Since the map is georeferenced, a global pose can be determined and output.

The EB robinos hardware component uses the current position included in the pose data to request the needed horizon data from the HD map provider via the connectivity provided by the cellular modem. From the data then made available, the ADASIS v3 horizon is obtained and provided to the **e-Horizon Reconstructor**. This module uses the pose data to extract a lane map from the horizon.

The **Lane Matching** module uses the motion as well as pose data and compares the lanes detected in camera images with the lane map provided by the e-Horizon Reconstructor module to achieve a localization relative to the map with lane-level accuracy. Since the map is georeferenced, a global pose can be determined and output.

Finally, the **Fusion** module receives the poses determined by the different algorithms as well as the motion and integrity data, and uses the available information to calculate the overall positioning solution that takes measurements from all the sensors into account. The resulting pose, motion and integrity data is output by the OBU to the subsequent components of the automated driving system.

6 Validation architecture

This section describes the approach towards the validation of the developed functions and systems and, in particular, the methodology to validate the proposed OBU. In ACCURATE, two tasks are defined to work on the design of this methodology, from the SW and HW perspectives:

- T3.5 Software in the Loop validation architecture design
- T3.6 Hardware in the Loop validation architecture design

The automotive safety validation processes are currently addressed in the extensively used automotive safety standard ISO 26262[1]. Its focus is on testing failures of electronic components by defining hazard analysis and risk assessment mechanisms. The emergence of increasingly complex ADAS (Advanced Driver Assistance Systems) which make use of perception-level sensors and data channels (V2X, GNSS) require the addition of complementary testing levels which address the safety of the intended functionality (SOTIF) event in the absence of HW or SW fault.

These aspects are covered by the standard ISO/PAS 21448[2], mostly focusing on SAE Level 2 functions. One of the novel aspects of this new regulation is about widening the concept of test results, where traditional pass-fail criteria needs to be statistically treated according to the environment and driving situation, as it does affect the capability of the function’s perception-layer to operate, in terms of performance, latency, range or availability.

Once the environment is a condition of the test itself, the validation process can no longer be simplified to laboratory stress-tests. The automotive industry has then been evolving during the last decade (2010-2020) to define methodologies to exhaustively measure functional safety in all driving situations. The approach has typically consisted on orchestrating large data collection campaigns, using instrumented vehicles which record large amounts of raw data of real world situations. Data is uploaded and processed to generate ground-truth descriptions of the scene (e.g. object-level information such as presence, position or velocities of other road participants). In HiL (Hardware-in-the-Loop) approaches, the Function-Under-Test (e.g. an emergency braking function, or lane keeping assistant) is then presented the raw data as it would receive it inside the vehicle, and its result is compared against the ground-truth to obtain measurements about its performance. The cost of labelling real data has become a major problem in industry, and vast investments are currently still spent on labelling technologies, recording campaigns, and HiL processes.

---

[1] Standard ISO 26262, Road vehicles – Functional safety, 2011.

[2] Standard ISO/PAS 21448, Road Vehicles – Safety of the Intended Functionality, 2019

Usually, the lifecycle of a FUT starts on design-level testing, with simulated components, environmental effects and models, in the early stages often called MiL (Model-in-the-Loop) and SiL (Software-in-the-Loop). The main challenges are still:

- to define a cost-effective strategy to manage the huge amounts of data to cover all possible situations the FUT will face in reality
- to measure the test coverage achieved so far with the executed tests
- to efficiently combine real and synthetic data in mixed HiL simulation test environments (where part of the system, environment or model is real, and part is virtual)
- to orchestrate the required computational effort to process data and to execute batch tests

These challenges are being tackled by the automotive industry, with various levels of success, via the following de-facto good practices:

- utilisation of HiL simulation platforms that help narrow down the test specification
- definition of ODD (Operational Design Domain) to formalize and structure the description of the environment conditions under which the FUT is supposed to operate correctly
- scenario-based testing, as a mechanism to standardise which situations need to be tested
- standardisation of input/output interfaces of components in the testing toolchain, aiming to gain interoperability across platforms, and vendors
- coordination of test methods (virtual testing, XiL, proving ground, field testing) and type-approval and regulation/certification bodies

Furthermore, current evolution into higher-level autonomous driving functions (SAE L3 and beyond) implies redefining the test concept itself in order to evaluate the vehicle itself while driving autonomously and not just a certain aiding function. In that sense, the validation process shall focus on treating the Vehicle-Under-Test (VUT) as the subject of the test, measuring its behaviour in terms of Safety, Comfort, Efficiency or Performance (compared to reference).

At the time of writing this deliverable (2021), different research projects and initiatives are paving the road towards declaring, specifying and standardising methodologies for autonomous driving validation. To name a few, the PEGASUS family of projects which have established the basis for scenario-based testing (including the projects VVM and Set-Level that aim to extrapolate the PEGASUS approach towards SAE L4-5 autonomous vehicles)[3]; the H2020 project HEADSTART that aims to add the Key-Enabling-Technologies

---

[3] https://www.pegasusprojekt.de/en/home

(KET) dimension[4]; or the ASAM e.v. simulation branch, which aims to create a family of standards to define interfaces for the automotive industry[5].

In ACCURATE, the main goal of T3.5 and T3.6 is to establish the SiL and HiL principles and frameworks. Nevertheless, considering the aforementioned topics in the automotive industry, a broader validation strategy is presented, which gathers the good practices and lessons learnt from these novel autonomous driving validation approaches. As the ACCURATE project runs in in parallel with some of the ASAM projects and the HEADSTART project, the presented architecture is inspired and based on their definitions and proposed strategies.

In particular, ACCURATE aims to produce an OBU, as a device which produces high-accuracy localisation capabilities to any vehicle, and thus can be seen as a SW-HW device that needs to be tested. The OBU, as defined in the architectural sections in this document, makes use of external source information, coming from the perception sensors (camera, LIDAR) and GNSS feeds. In that sense, the OBU will be treated as a FUT (Function-Under-Test) and tested using HiL simulation set-ups. Test requirements and parameters for localisation components as defined in the HEADSTART project (public deliverable “D2.2 Extension of the Common Methodology for the HEADSTART Key Enabling Technologies”) will be used as guidelines in ACCURATE.

The challenge will be to adequately address the need to model, simulate and feed the positioning information into the HiL simulation set-up.

6.1 XiL Validation – General approach

The general XiL architecture of ACCURATE is depicted in Figure 12. The architecture includes the following stages:

- Scenario database generation
    - From expert knowledge (from use cases)
    - From processing real data (data recording campaigns, scenario mining)
- Scenario allocation for tests
    - From “functional” to “concrete” scenario (i.e. OpenScenario)
- Test specification and test framework set-up
    - With definition/selection and interconnection of virtual and real components

---

[4] https://www.headstart-project.eu/
[5] https://www.asam.net/

- Test results analysis
    - Harmonisation of test results across different platforms
    - Feedback to scenario searching

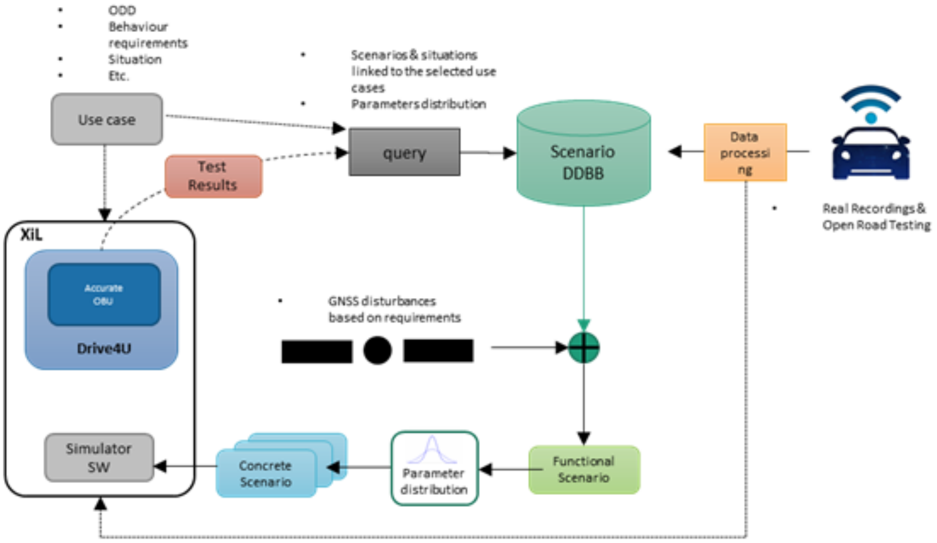


Figure 12: ACCURATE validation architecture.

The central part of the architecture is the existence of a platform that manages the pipeline, and the interfaces between steps. Generally speaking, the process starts with a database of scenarios (Scenario DDBB in the figure), which contains a structured description of scenarios potentially of interest to test the Function-Under-Test. The scenarios are created using two possible sources:

- **Expert knowledge**: considering the function to be tested, the operational design domain, and other requirements, a scenario designer user can create a description of the scenario.
- **Real processed data**: scenario mining consists on a series of processing steps which analyze the real data from instrumented vehicles and with the help of semi-automatic tools, generate descriptions of the observed scenarios.

There is a large debate in the industry about which approach is better, and, as usual, a mixed approach seems to include benefits from both options. On the one hand, expert-based scenarios are focused, as the requirements or test interests can be imposed explicitly (e.g. following safety regulation specifications, such as the “ALKS Regulation UNR157” scenarios[6]), while data recording campaigns suffer from “open world” circumstances which include highly complex and highly unpredictable situations to be captured in real roads. On the other hand, the realism that can be achieved with manually built scenarios is in question, and

---

[6] ECE/TRANS/WP.29/2020/81 - https://undocs.org/ECE/TRANS/WP.29/2020/81

thus safety validation based on just hand-written scenarios will not be sufficient for most regulatory contexts.

A mix between them imply the utilisation of expert-based scenarios as a basis or as a set of restrictions, concretized and detailed by using as much real information as possible, either from scenario mining from data campaigns (e.g. averaging velocities at certain road stretches from equipped vehicles), or via the utilisation of realistic models either for agent behaviour (e.g. ego-vehicle dynamics, road physical properties), perceptual systems (e.g. physical-based LIDAR simulators), or environmental conditions (e.g. multi-path GNSS simulators).

Considering the needs and limitations of the ACCURATE project, the proposal is to use existing scenarios, as much as possible to minimize the costs and time needed to prepare the evaluation stage, and enrich them with the required level of detail about the OBU features under test.

In either case, the scenario is materialized in the database as a “functional” scenario, as a structured description of the temporal evolution of the scene, including static, quasi-static and dynamic elements (see 5-layer model as defined by PEGASUS and adopted by ASAM standards); plus a set of satellite information such as the parameter distributions.

As proposed in HEADSTART, the description of this “functional”-level database can be made semantically typed by using ontologies that govern the terms used, linking to standardise and universal concepts, along with possible relations between the terms (e.g. objects, events or actions), such that additional reasoning capabilities are enabled at the database. Candidate technologies to host this database are graphical databases (e.g. Neo4j with neosemantics plugin[7]) which map scenario descriptions as graphs, following the recommendations of the recent ASAM OpenXOntology[8] project.

Querying the database can be done using a dedicated web application that retrieves results and displays possible scenarios of interest for the test engineers. When selected, according to the desired function-under-test, the scenario suffers a “concretization” process, which converts the “functional” (high-level, semantical) description of the scenario, retrieves the parameter distributions, and determines which specific values can be put together to create consistent and reasonable test cases. In this process, additional information can be added to incorporate test parameters, such as desired GNSS disturbances or any other condition that wants to be taken into consideration during the test.

---

[7] https://neo4j.com/labs/neosemantics/4.0/
[8] https://www.asam.net/project-detail/asam-openxontology/

At this stage, before the tests can actually be prepared, the test methods need to be evaluated themselves. Their capabilities need to match the requirements of the tests. For example, a virtual environment which is not able to simulate or inject GNSS disturbances into the simulation can not be used to test the OBU robustness. Similarly, if a SiL set-up does not have a mechanism to simulate a LIDAR or camera sensor, the SLAM component of the OBU can not be tested.

The materialization of the “concrete” scenario is done using the well-known ASAM standards OpenDrive (for static road topology and geometries description) and OpenScenario (to describe the dynamic part of the scene). These files follow execution semantics and determine how the scene should evolve. In virtual environments, that implies that a simulation engine needs to read the OpenScenario files and run the different involved components (e.g. agents, ego-vehicle, environmental models) aiming to reproduce the specification as closely as possible. For non-virtual domains, the specification needs to be followed either by the test engineering team or by the infrastructure supporting the test (for example to reproduce the desired levels of signal perturbation, participation of other vehicles or Vulnerable Road Users, etc.).

Once the test specification is in place, and the test method selected (or test methods, to be executed sequentially), the test can actually happen. In ACCURATE the test methods under consideration are mostly XiL simulation, which includes all necessary and available levels of XiL, such as Software-in-the-Loop and principally Hardware-in-the-Loop. Most likely, the tests will first happen at a purely virtual framework, where all components are simulated, including the OBU itself, the environmental conditions and traffic participants, the in-vehicle systems (e.g. perception and localisation), and potentially, the infrastructure elements that determine the characteristics of GNSS signals consumed by the vehicle.

In a second stage, the software components will be replaced by HW components, such as the OBU prototype to be built in ACCURATE, and connected to a physical HiL simulation framework, which feeds the OBU with either simulated or real data, as it were functioning in real conditions in a real vehicle. Depending on the type and scope of the test, the HiL simulation framework may need to be bridged to additional SW or HW components, which can execute additional functions (e.g. perception algorithms to detect obstacles, or specific drivable areas) or inject real-time signals required to evaluate the performance of the localisation components.

The ultimate goal of the testing stage is to produce test results. Following the ongoing discussions of the ASAM Test Specification group, and some of the proposals of the HEADSTART’s “D3.4 Harmonisation proposal of test results”, test-related data can be described as follows:

- **Test parameters/specifications**: variables whose value determines the behaviour of the function-under-test, organized under categories such as Safety, Comfort, Efficiency, Tactical, etc.

- **Test requirements**: values, ranges or thresholds defined for the test parameters, which will define whether the test has succeeded or failed (a Boolean or statistical pass/fail criteria can be defined)
- **Test measurements**: data obtained during the test execution, sampling the status of the involved components at a given target frequency, e.g. positions, velocities and other magnitudes of vehicles (either at the real stage or within the simulation framework).
- **Test results (KPIs)**: the list of results where the test parameters are compared to the test requirements and thus a test report is produced. Frequently known as KPI (Key Performance Indicator) computation.

The execution of test has, therefore, to be orchestrated to have a clear and structured definition of test parameters and requirements, and then coupled with a number of adjacent sub-systems which are actually able to gather test measurements, and processing capabilities to compute the desired KPIs (either online or offline).

Virtual platforms will typically use real-time processing capabilities to query the simulation engine about the test measurement via programmatic APIs, and then external components can consume these data and produce the KPIs (e.g. number of collisions, average time-to-collision times, lateral acceleration values, fuel consumption, etc.).

6.2 Capabilities of the XiL testing

XiL (X-in-the-loop) is the extended concept of testing a certain SUT (System Under Test, where System can be a certain Device, Function or Vehicle, depending on the level at which the testing is applied) outside its real operational environment by replacing part of the required systems by other systems which produce/consume equivalent signals from/to the ECU and which offers logging and evaluation capabilities.

In the context of automation, XiL is a novel term, which aggregates the different stages of testing such as MiL (Model-in-the-Loop), SiL (Software-in-the-Loop) and HiL (Hardware-in-the-loop), with other variants such as DiL (Driver-in-the-loop), or ViL (Vehicle-in-the-loop).

In HiL simulation a physical SUT (sometimes called ECU, Electronic Control Unit) is connected to a HiL platform to communicate input/output signals. The HiL platform “fakes” (thus the term “simulation”) the real interfaces the SUT will have in its operational domain. For instance, a SUT which automatically determines the presence of objects from images obtained through camera sensors. In this context, and in practice, the HiL platform can be of two types: (i) a player or raw recorded scenes, (ii) a simulation engine which produce virtual scenes

There are two main execution types of XiL:

- **Open-loop**: the ECU only receives information from the XiL platform whose role is simply to feed the ECU with data streams and capture its behaviour for logging and test result composition.
- **Closed-loop**: the ECU interchanges information with the XiL platform, which, based on the received information, may need to real-time modify or adapt the streamed data to the ECUs.

6.3 Input/Output & Data formats

This section provides information about standardized data formats and interfaces, including their purpose, working principles, and ongoing standardization activities. Most of the further mentioned standardization processes are organized by ASAM e. V. (Association for Standardization of Automation and Measuring Systems).

Data formatting is one of the major topics under the ASAM e.V. simulation standards. The family of OpenX standards is devised to cover most of the domains involved in testing and evaluation of ADAS and AD, such as “sensor data labelling”, “scenario generation”, “scenario search”, and “test description and simulation”.

In ASAM, a well-known family of standards are being developed in parallel, aiming to produce a set of recommendations and data formats compatible, harmonised and interoperable (see Figure 4). In addition, existing and consolidated standards from other domains are currently under analysis for further utilisation or adaption to current needs of SAE-L3 and above.

- **ASAM MDF (Measurement Data Format)** – specification of data format as container for sensor data.
- **ASAM OpenLABEL** – specification of metadata format for labelling sensor data at object level and semantic level (e.g.: actions, events, relations).
- **ASAM XIL API** – specification of interfacing for XiL frameworks.
- **ASAM OpenDRIVE** – specification of road topology description for HD maps.
- **ASAM OpenSCENARIO** – specification of scenarios for automotive testing.
- **ASAM OpenODD** – specification of Operational Design Domain for (autonomous driving) functions under test.
- **ASAM OpenXOntology** – specification of domain models and ontologies to harmonize concepts from all other OpenX standards.
- **ASAM OSI (Open Simulation Interface)** – specification of interface between simulation engines and components.

- **ASAM ATX (Automotive Test Exchange format)** – specification of standardised XML format to enable exchange of test data between different systems.
- **ASAM ODS (Open Data Services)** – specification of mechanisms for persistent storage and retrieval of testing data.

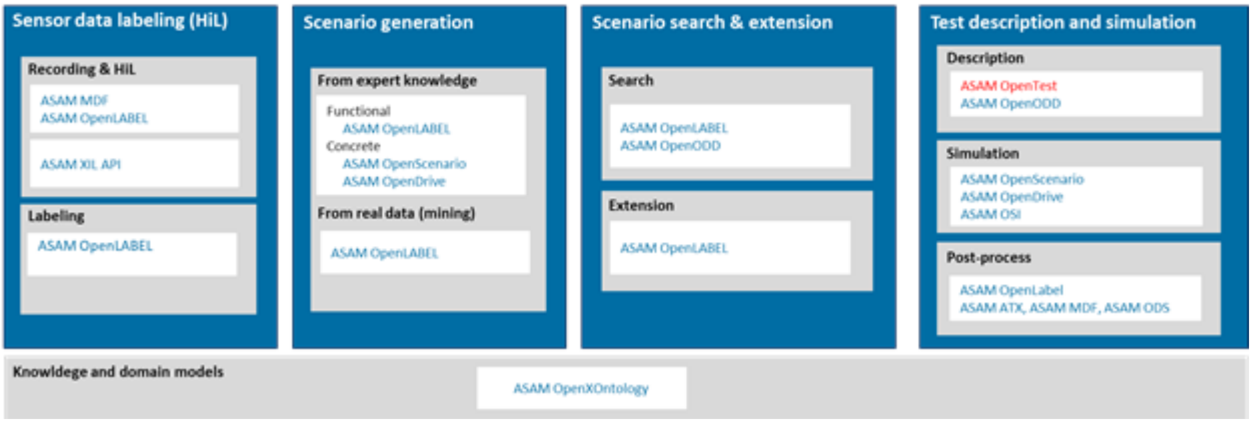


Figure 13: ASAM OpenX standards (simulation branch).

In ACCURATE, these data formats are identified to be relevant at different stages of the proposed methodology. At the scenario search and retrieval from databases, ASAM OpenSCENARIO® and ASAM OpenDRIVE® files will be obtained or generated, along with the necessary parameter distributions, and optionally, additional labels or scene tags in the form of OpenLABEL files.

At query time and also for allocation of scenarios, ASAM OpenODD can be utilised to select which scenarios are relevant, and also to prepare the test cases.

During test execution, especially for virtual testing, ASAM OpenSCENARIO® and ASAM OpenDRIVE® files are required along with other adjacent files for rendering or visualization. Test results and measurement data can be exported as MDF files and ATX exchange formats (at the time of writing this deliverable, the ASAM Test Specification group is in its earliest stages of discussions to define its goal and purpose and whether it is necessary or if its scope is already covered by existing standards).

Finally, large-scale consumption of test data can benefit from using principles from ODS (Open Data Services), where a test data management system built on top of a test system manages the produced test measurements or calculated data.

6.4 XiL Toolchain

The architectural principles of ACCURATE include the utilisation of a middleware layer that abstract the execution of the components, and thus makes possible the seamless interconnection of multiple sensors and devices, real or virtual. In particular, ACCURATE will make use of the RTMaps middleware software, as a flexible multi-sensor application framework. It does work as a bridge between real sensors inside a vehicle and the SW components of the OBU and other elements of the vehicle. At the same time, it includes recording capabilities and playback functionality, which directly connects to the HiL simulation principles described in this section.

For instance, it will be possible to develop OBU internal functions as RTMaps components and then run them on any computer or HW device inside a real vehicle, or in a HiL simulation set-up, feeding it with real recordings captured with an instrumented vehicle (with streams of video, point clouds, and GNSS data), or simulated data from a signal simulator (see Figure 14).

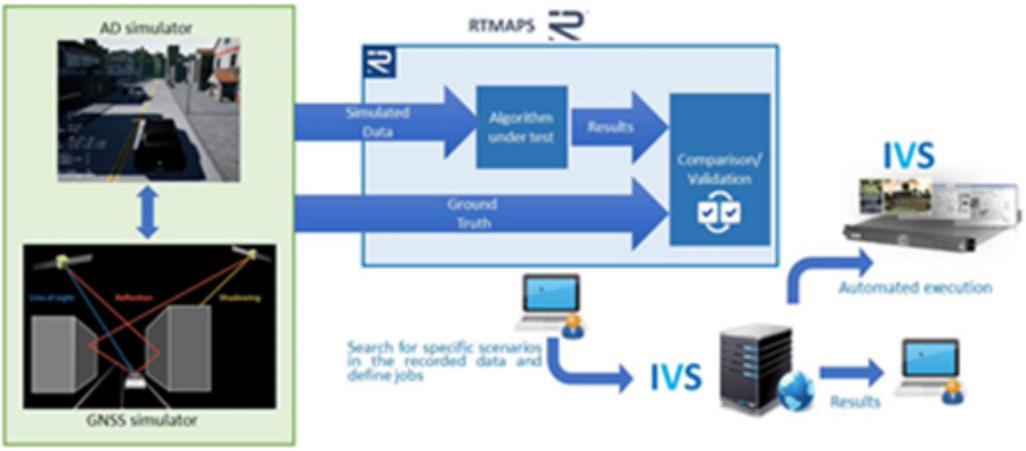


Figure 14: ACCURATE HiL simulation set-up using RTMaps.

The playback functionality (as illustrated in Figure 15) is particularly useful for development and debugging purposes, as the same reference data recordings can be used by all members of the consortium as a benchmark, simplifying interfacing and integration tasks, and accelerating prototype building.

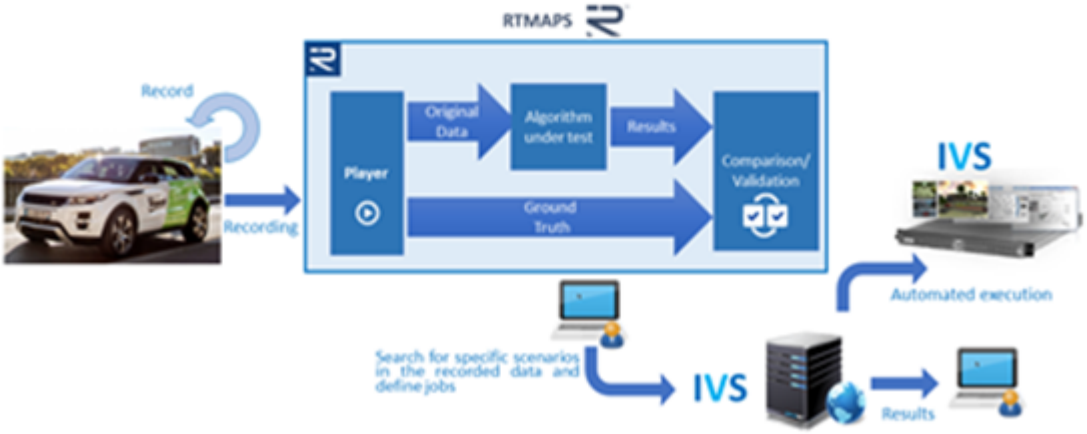


Figure 15: ACCURATE playback testbed using RTMaps.

6.5 Environment Simulation

Simulation engines are key to provide fast and early test stages of FUT (Functions-Under-Test). Mixed test approaches, as explained along the chapter, imply the substitution of certain components, agents or conditions by their virtual counterparts.

Simulation can then be useful to span ranges of parameters otherwise difficult or costly to capture with real data. For a valid testing strategy, virtual components must be validated themselves, in the sense their modeling capabilities meet the expected fidelity requirements.

In ACCURATE, the Function-Under-Test are the OBU functions: localisation and positioning functionalities. The simulation strategy must then define which components are virtualized, their level of fidelity, and their interfaces to cooperate with real components under a test platform (e.g. HiL simulation framework). In general, the following simulated pieces are considered:

- Perception sensors: virtual models of cameras (RGB, IR), LIDAR, RADAR.
- Positioning devices: GNSS signal simulators.
- Environment: virtualized cities and roads where the action happens, including illumination and weather conditions.
- Vehicle and functions: virtual models simulating the actual driving task or a specific function.

If all these components are left as virtual, then the approach is usually defined as virtual testing, otherwise understood as a variation of Hardware-in-the-Loop.

At the point of writing this deliverable, the most suitable options are:

- **HiL simulation, with GNSS signals recorded from real driving sessions**; specifically devoted to test the TCU functionality (localisation, first level of data fusion). The expected TCU HW will be connected to a HiL station, through standard interfaces, to playback the recorded signals, and gather the TCU outputs. Ground-truth or expected signals need to be determined (e.g. labeled or known before hand) to compare the signals in a KPI computation component.
- **SiL and HiL simulation for perception fusion**; where inputs from the TCU are either fully simulated (using a mock-up generator) or recorded from real drives. The Positioning component receives (via the SiL or HiL setup) also virtual perception inputs (e.g. sequence of point clouds from virtual LIDAR and/or detected lanes from camera sensor), and the computes the second level data fusion. Depending on the simulation engine, these perception signals will be fully virtual or also recorded and played back. The difference between the SiL and HiL approach is that the OBU positioning

component can be first tested while being a SW component, run in any server at the time of testing (SiL), or when already deployed into the target embedded PC and thus connected to the mock-up signals via a HiL platform.

Considering these options, testing and validation implies the utilisation and preparation of the following components:

- **Simulation engine**: a framework that virtualizes the environment, agents and potentially also have APIs to connect with user-defined virtual models of sensors, GNSS signals, and bridges to other platforms.
- **HiL framework**: a platform that can real-time receive and forward signals from different components, including the simulation engine and the device under test (using standard interfaces and cabling). In addition, the HiL framework can be sufficiently powerful to also provide real-time computation capabilities, such as KPI computation, data log recording, etc.
- **Execution middleware**: to simplify the execution of SiL and HiL test runs, a middleware layer is fundamental. This layer abstracts the design, development of SW functions from the underlaying HW, by enforcing the developers to encapsulate functions with standard interfaces, and taking care of their seamless execution in different platforms (e.g. testing server and target embedded device). Current options for ACCURATE are RTMaps (main product of Intempora) and ROS (open source Robotic Operating System), which will be considered along the project according to their compatibility levels with the developments and test platforms.

Although out of the scope of this section, the following list briefly summarizes some well-known simulation engines and to their capabilities:

- **NVIDIA Drive Constellation simulation platform**: composed of two servers running the simulation and the computation to operate as a HiL system, and making use of DRIVE AGX Pegasus AI car to simulate the driving functions.
- **Simcenter Prescan**: professional physics-based simulation platform used for the development of ADAS and automated vehicle functionality.
- **CARLA:** open source, growingly popular driving simulation engine. Compatible with OpenScenario 1.0 and OpenDrive 1.6, contains autopilot models, sensor models, and photorealistic rendering using the Unreal engine. Bridges exist with ROS, SUMO traffic simulation, and fully customizable Python API.
- **IPG Carmaker:** includes a complete model environment comprising an intelligent driver model, a detailed vehicle model and highly flexible models for roads and traffic. Special actions, such as

driver activities or system interventions can be modelled in CarMaker using manoeuvres. Malfunctions, which also occur in real-world test runs can be triggered precisely via events.

- **VIRES VTD:** is a toolkit for the creation, configuration, presentation and evaluation of virtual environments in the scope of road and rail-based simulations. It covers the full range from the generation of 3D content to the simulation of complext traffic scenarios, and finally, to the simulation of either simplified or physically-driven sensors. It is used in SiL, DiL, ViL and HiL applications and may also be operated as co-simulations including 3rd party or custom packages.

As a side remark, all these simulation engines include some sort of GNSS sensors, though modelled quite simplistic: typically, longitude, latitude and altitude of objects (including the ego-vehicle) are captured from the simulation and a noise-addition API function permits the user to add perturbations to the signals. However, for more complex and realistic effects, such as multi-path signal effects, and full GNSS constellation simulation are only available in specialized SW packages, such as Spirent's Sim3D, though without the required capabilities of a full driving simulation engine. In addition, some HiL frameworks are also compatible or coupled with HW GNSS signal simulators, which then effectively enables their utilisation in mixed testing approaches. To name a few: Rhode&Schwarz SMBV100B vector signal generator[9], Syntony GNSS simulator[10], Orolia GPS/GNSS simulator[11].

---

[9] https://www.rohde-schwarz.com/es/producto/smbv100b-pagina-de-inicio-producto_63493-519808.html

[10] https://syntony-gnss.com/products/gnss-simulator/

[11] https://www.orolia.com/products/gnss-simulation/gpsgnss-simulators

7 Conclusions

This document contains the agreed reference preliminary architecture of the ACCURATE OBU. It does contain a consensus view on the ACCURATE OBU functional design, which includes a breakdown of components in the form of functional units, with declared input and output interfaces.

Two major perspectives are provided: HW and SW. For each perspective, possible implementation considerations are provided, such as specific HW platforms to use, protocols to be adopted, and SW frameworks to install. As a general remark, a middleware layer is selected to abstract the implementation of the necessary multi-sensor and multi-equipment system. The RTMaps middleware is selected and intended to provide the necessary abstraction to facilitate the development stages of ACCURATE.

In addition, this deliverable also contains a proposal for the evaluation mechanisms to be adopted for testing the ACCURATE OBU. The proposal gathers current practices towards SAE L3+ testing and evaluation approaches, including concepts like scenario-based testing, XiL simulation, and standardisation of measurement data and test results.