PROJECT "LOCUS": LOCalization and analytics on-demand embedded in the 5G ecosystem, for Ubiquitous vertical applicationS

Grant Agreement Number: 871249
(https://www.locus-project.eu/)

DELIVERABLE D2.4

"System Architecture: preliminary version"

Deliverable Type:	R
Dissemination Level:	Public
Contractual Date of Delivery to the EU:	31/07/2020
Actual Date of Delivery to the EU:	09/08/2020
WP contributing to the Deliverable:	WP2 – Use cases, Requirements, Security, System Architecture
Editor(s):	Incelligent, Kostas Tsagkaris
Author(s):	CNIT, Stefania Bartoletti, Nicola Blefari Melazzi, Andrea Conti, Domenico Garlisi, Danilo Orlando IBM, Joseph Antony, Gabriele Ranco Incelligent, Athina Ropodi, Aristotelis Margaris, Nikos Psaromanolakis NEC, Giuseppe Siracusano

	NXW, Giacomo Bernini, Elian Kraja, Mario Salinas Orange, Sana Ben Jemaa SAMS, Mythri Hunukumbure, Tomasz Mach TEI, Stefano Stracca, Marzio Puleri VIAVI, Takai Eddine Kennouche, Chris Murphy, Howard Thomas UMA, Sergio Fortes, Eduardo Baena, Antonio Tarrías, Raquel Barco VIAVI, Takai Eddine Kennouche, Chris Murphy, Howard Thomas
Internal Reviewer(s):	CNIT, Andrea Conti IBM, Joseph Antony Orange, Sana Ben Jemaa
Short Abstract:	The goal of this deliverable is to detail the reference architecture of LOCUS, compliant with the system requirements and use-cases defined in Task 2.1. As a preliminary version, it describes the main functional blocks and subcomponents, related localization analytics and other functions, as well as provides an initial view of the System and its deployment options, to be finalized at the end of Task 2.3.
Keyword List:	LOCUS Reference Architecture, Localization, analytics, LOCUS System View, Deployments aspects.

Executive Summary

LOCUS aims at providing a unified and generalized localization analytics platform that will offer location-based analytics relying on machine learning and/or deep learning and that is able to support a wide variety of applications defined within the project. This deliverable elaborates on the reference architecture of LOCUS based on the LOCUS use cases defined in Deliverable 2.1 and their high-level requirements, as well as system-level, and security and privacy requirements. Following a short review of the state of the art approaches, the deliverable presents the major functional blocks and defines the various functions of the LOCUS platform, indicative applications related to the suggested use cases, and services to be exposed at the application layer -external to the LOCUS Platform. A comprehensive view of the LOCUS system architecture is also presented along with the deployment aspects through the LOCUS management and orchestration functions on top of a virtualized infrastructure spanning across edge and core locations.

Table of Contents

List of Abbreviations

ABBREVIATION	FULL NAME
3GPP	3rd Generation Partnership Project
5G	Fifth generation technology standard for cellular networks
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
AMF	Access and Mobility Function
AoA	Angle of Arrival
API	Application Program Interface
ARIMA	AutoRegressive Integrated Moving Average
ASN.1	Abstract Syntax Notation
BeFEMTO	Broadband Evolved FEMTO Networks
BS	Base Station
BSS	Business Support System
CID	Cell ID
CM	Configuration Management
CNF	Containerized Network Function
CNN	Convolutional Neural Nets
CRUD	Create, Read, Update, Delete
CSAR	Cloud Service ARchive
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DL	Deep Learning
DoA	Difference of arrival
e-CID	Enhanced Cell ID
eLCS	Core LoCation Service
EMF	Electric and Magnetic Fields
FM	Fault management
GMLC	Gateway Mobile Location Centre
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HCS	Highly Connected Subgraphs
HTTP	Hypertext Transfer Protocol

IaaS	Infrastructure as a Service
IoT	Internet of Things
ISG	Industry Specification Group
IWD	Inverse Weighted Distance
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
KPI	Key Performance Indicator
LCM	Lifecycle Manager
LDR	Location Deferred Request
LEN	Localization Enabler
LMF	5G Core Location Management Functions
LPI	Local Polynomial Interpolation
LSTM	Long short-term memory
LTE	Long-Term Evolution
MANO	Management and Orchestration
ML	Machine Learning
MO-LR	Mobile Originated Location Request
MT-LR	Mobile Terminated Location Request
NAS	Non-access Stratum
NEF	Network Exposure Function
NFV	Network Function Virtualization
NG-RAN	Next Generation Radio Access Network
NI-LR	Network Induced Location Request
NN	Nearest Neighbours
NSD	Network Service Descriptor
NSD	Network Service Descriptor
NSE	New Services
NSM	Network Service Mesh
OID	Object Identifier
OSM	Open Source MANO
OSS	Operations Support System
PCA	Principal Component Analysis

PLMN	Public Land Mobile Network
PM	Performance Management
PNFs	Physical Network Functions
PRS	Positioning Reference Signal
QoS	Quality of Service
RBF	Radial Basis Function
REST	REpresentational State Transfer
RNN	Recurrent Neural Network
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SBA	Service Based Architecture
SBI	Service Based Interface
SINR	Signal to interference and noise ratio
SNM	Smart Network Management
SNMP	Simple Network Management Protocol
SoTA	State of The Art
SRS	Sounding Reference Signal
SSB	Single-Sideband modulation
SVM	Support Vector Machine
SQL	Structured Query Language
TDoA	Time difference of Arrival
ToA	Time of Arrival
TOSCA	Topology and Orchestration Specification for Cloud Applications
TPS	thin plate spline
t-SNE	t-Distributed Stochastic Neighbour Embedding
TTC	Time To Collision
UC	Use Case
UDM	Unified Data Management
UE	User Equipment
UMAP	Uniform Manifold Approximation And Projection

UWB	Ultra-Wideband
VIM	Virtualized Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
VNFFG	VNF Forwarding Graph
VNFM	VNF Manager
VRU	Vulnerable road user
WP	Work package
XML	(e)Xtensible Markup Language
YAML	YAML Ain't Markup Language
ZSM	Zero-touch network and Service Management

Table 1.1: Abbreviation List

Table Index

1 Introduction

1.1 Introduction and Audience

Deliverable 2.4 “System Architecture: preliminary version” aims to offer a preliminary definition of the LOCUS platform. This deliverable is part of Task 2.3 “System Architecture Specification and Implementation” and its finalized version (Deliverable 2.5) is due in month 20 of the project. As this is a public deliverable, it is intended for the LOCUS consortium members, the EU commission, as well as general public.

Its main objective is to elaborate on the reference architecture of LOCUS taking into account (a) the LOCUS use cases (UCs) as defined in Deliverable 2.1 [1] and their high-level requirements, so as to be supported by the LOCUS platform derived from them, (b) the security and privacy requirements that are related to them and is part of the work done in Task 2.2, and, (c) system-level requirements, given the platform’s expected ability to provide advanced localization analytics in order to enable both Smart Network Management applications, as well as Vertical sector applications through exposing services to 3rd parties.

Following a short review of similar architectures, the deliverable starts with the high-level description of the major functional blocks and then goes into more detail, defining the various functions of the LOCUS platform and describing their use, possible processing and analytics, expected inputs and outputs. These functions can be grouped in multiple categories, specifically data collection, data management, analytics functions (i.e. analytics that may be internal but also be exposed as services), machine learning (ML) and Artificial Intelligence (AI)/deep learning (DL) functions, security and privacy functions, as well as management and orchestration functions. The deliverable also provides comprehensive examples of how various functions can be decomposed into other functions instantiated and chained together, creating -in a way- function pipelines for the per UC expected LOCUS results. In addition to this, although outside of the LOCUS platform per se, the indicative application descriptions that cater to the defined UCs are described and the service exposed for these applications are defined. This also provides a first attempt of the system view and the expected deployment options.

Lastly, it should be mentioned that this deliverable is based on the first results and requirements derived from the work that is currently in progress within the LOCUS project and therefore is preliminary and can be subject to changes, additions and consolidations so as to be in line with the overall project progress.

1.2 Document Structure

This Deliverable is structured in 9 sections. Following the Introduction (Section 1), the following sections are included:

- Section 2 presents a short review of the state-of-the-art (SoTA) work related to the LOCUS architecture;
- Section 3 describes the high-level Functional Architecture of LOCUS;
- Section 4 details the Functional Architecture;
- Section 5 describes the App Layer Functions;
- Section 6 defines the Services exposed by the LOCUS platform to the application layer;
- Section 7 presents the LOCUS-related Standardization Activities; and
- Section 8 presents preliminary LOCUS System Architecture and Deployment options.

The last section (Section 9) summarizes the conclusions and next steps towards the final deliverable.

2 Short Review of SoTA Architectures

In the 5G context, communications are characterized by a wide variety of use cases leading to an increase of several orders-of-magnitude of connected devices. As a consequence, current mobile communication networks will evolve into an ultra-dense heterogeneous network [1] [2]. In order to fulfil the resulting data rate demands, location information plays a role of primary importance so much so that communications in the next 5G network are labelled as location-aware. It is important to highlight that starting from Release 16, 3GPP has begun to address the integration of localization primitives into the 5G system architecture. To this purpose, the 3GPP Technical report 23.731 [3] describes the 5G Core LoCation Service (eLCS) architecture, and discusses relevant security issues, including methods to address data eavesdropping and mobile location tracking.

The main strength of the LOCUS architecture is that it represents a natural, complete and integrated solution for location services within this dense and highly connected operating scenario that comprises mobile and/or steady heterogeneous nodes bearing a huge amount of information. It is important to underline that existing location systems/architectures exploit only in part the information provided by the entire set of heterogeneous sources considered in the LOCUS platform. In fact, most of them only merge/process data from specific subsets of sources which can be, for instance, satellite location systems, intelligent transportation systems, or the network infrastructure [4][5]. Just to provide some examples, the authors of [5] propose a network-centric architecture for outdoor location and tracking of mobile devices in cellular networks. Such architecture relies on the joint exploitation of the measurements collected by the base stations, the angular sector covered by the antenna, and open map data.

In the same context of integrating localization information as well as other context variables, sources of information outside of the cellular network ones, such as social data have been considered. To do so, the integration between those sources (that could be provided by sources of information external to the cellular network) shall be integrated into the general architecture of the system. Previous works have tried to address such a challenge. The European FP7 project Broadband Evolved FEMTO Networks (BeFEMTO) focused on efficient solutions for LTE-A femtocells management [6]. In this context, it identified localization as a key enabler for the cellular management. It also proposed an extension to the 3GPP architecture by adding core and local entities to minimize the related traffic into the operator's core and backhaul. An architecture for the integration of both internal and third-party location sources into cellular management, considering centralized, distributed and hybrid approaches was presented in [7]. This work also focused on minimizing the impact of the use of the location fluxes of information (per each User Equipment-UE and updated regularly) on the cellular network backhaul and core networks. It also included some discussion on the flows to exchange positioning and associated network management

information between the different elements of the network in LTE for both integrated solutions in the management/control plane of the network as well as by “over-the-top” solutions. Furthermore, the integration of different sources of context information and related challenges have been discussed in [8], where a general scheme was defined. Additionally, a very high-level framework for the use of social-based data into cellular management activities was discussed in [9].

In [10], the information from heterogeneous sensors is fused for location services. The conceived system architecture is distributed and deployed in the context of intelligent transportation systems with the objective of increasing the accuracy provided by satellite-based location systems. Moreover, it considers vehicles endowed with suitable sensors whose measurements are merged and integrated with those provided by other vehicles or a specific communication infrastructure. In [11] and [12], the authors develop a framework for the design of scalable network localization and navigation algorithms exploiting information from heterogeneous devices in the context of Internet of Things. Ultra-Wideband (UWB) radio technology represents the key technology of the architectures developed in [13], whose field of application ranges from domotics to public transportation/automotive environments. The choice of UWB technology is motivated by the fact that it offers highly precise localization capabilities [14][15]. However, these platforms are ad hoc solutions and, more importantly, are not part of the network infrastructure. Another example of distributed location architecture is described in [16], where a dense network of cooperating nodes is designed for indoor person detection and tracking. This solution is based on Radio Frequency Identification and UWB technologies which are more effective for indoor environments. In [17], the authors conceive a platform that provides a “Location as a Service” by merging both Galileo and the European Geostationary Navigation Overlay services with other available location enablers. The Global Navigation Satellite System (GNSS) represents the main source of location information for the architecture proposed in [18], where the position estimates provided by GNSS are suitably integrated with in-situ data and displayed with maps and geo-spatial space information in a timely, seamlessly integrated, secure and user-friendly way. Finally, the system designed in [19] is aimed at locating isolated victims in the case of natural or man-made disasters by means of the fusion between the position data obtained by European GNSS and those provided by Digital Cellular Technologies.

3 High-level LOCUS Architecture

A preliminary high-level architecture for the LOCUS platform was presented and described in D2.1 [1]. This chapter aims at providing a summary of this initial architecture definition, as an introduction to the next chapters where more details on the various LOCUS capabilities, functions and applications are provided.

LOCUS provides a platform for localization analytics to be offered and exposed to Smart Network Management and New Services/3rd party applications, and makes use of 3GPP technologies combined with analytics and Machine learning/Artificial Intelligence (ML/AI) techniques.

Figure 3.1 shows the main LOCUS functional blocks as defined in D2.1 [1], providing a high-level view of how the various functions, data repositories, management components relate with each other within the LOCUS platform as enabled by the LOCUS APIs.

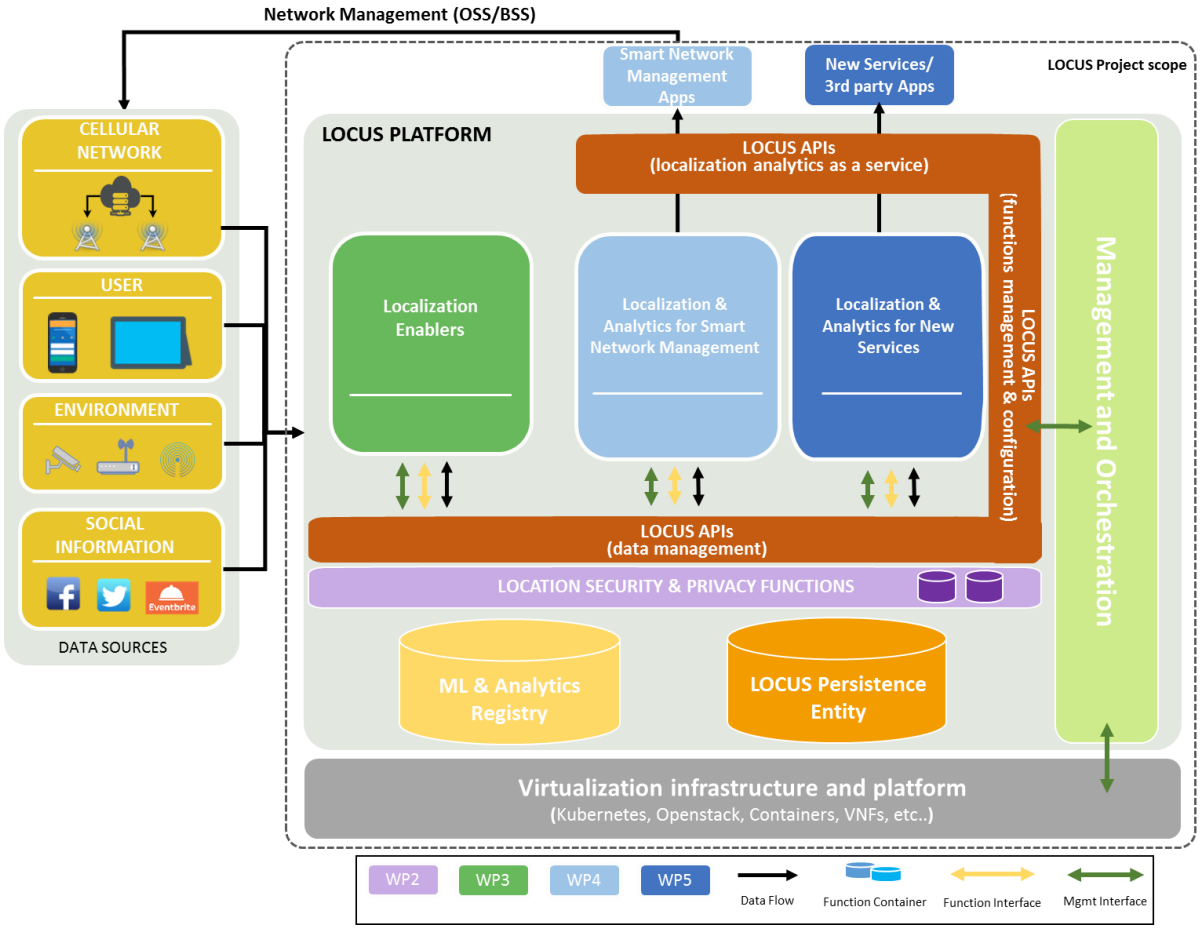


Figure 3.1 High-level LOCUS Architecture

A brief description of each of these components is provided in the table below.

Table 3.1 LOCUS Functional Blocks

Functional Block	Description
Localization Enablers (LEN)	This functional block is responsible for providing the user device location data (e.g. coordinates, velocity, direction) needed by the other LOCUS components. It utilizes data coming from various 5G network and other sources, i.e. network data, user device, external and social data, outside the LOCUS system and provides an estimation of the user device location. This block allows for continuous localization and tracking with varying levels of granularity of the user devices with high positioning accuracy, so as to distinguish between different users, vehicles, and targets within an area. For this reason, it employs analytics, data fusion and ML mechanisms, to achieve high accuracy, ensure data reliability for the detection and mitigation of attacks against location data and correction of the aforementioned data. The information acquired from the external sources, as well as the expected positioning-related output is stored in the *LOCUS Persistence Entity*.
Localization & Analytics for Smart Network Management (SNM)	This functional block is responsible for the use of Analytics and ML mechanisms in conjunction with the position-related information provided by the *Localization Enablers*, so as to offer localization analytics services related to KPI monitoring and knowledge building, network planning, management, optimization and diagnostics purposes. The results are exposed outside of LOCUS through the *LOCUS APIs* and can be used by other external Smart Network Management applications and, given availability of the appropriate APIs, can be applied towards OSS/BSS. Intermediate results can be stored internally within the LOCUS platform and re-fed to the various subcomponents, enabling the internal communication of the various subcomponents.
Localization & Analytics for New Services (NSE)	Similar to the previous functional block, the location-related information is utilized in order to expose through the *LOCUS APIs*, Analytics and ML-derived results for localization analytics as a service towards vertical and third-party applications. Based on the identified use cases in D2.1 [1], these analytics and ML results relate to people mobility, flow monitoring and self-driving objects.
LOCUS Persistence Entity	The *LOCUS Persistence Entity,* i.e. the LOCUS data store, is responsible for the storage of all relevant data, including data collected from sources outside of LOCUS, as well as analytics/ML results and metadata. It enables both batch ingestion as well as data streaming functionalities for data coming from outside the LOCUS platform or the

	various LOCUS functional components. All of these data are exposed within the LOCUS system through the *LOCUS APIs.*
ML & Analytics Registry	The *ML & Analytics Registry* includes analytics functions and ML models (e.g. clustering, anomaly detection, classification, regression, and time-series modelling). These functions of the system shall be employed by more than one component for the derivation of localization analytics that will in turn be exposed through the LOCUS APIs to applications external to the platform.
Security & Privacy Functions	The location security & privacy functions relate to all the data privacy and security aspects, including authentication and advanced cryptographic techniques for all network interfaces (homomorphic encryption, secure multiparty computation, and secure conditional sharing techniques, data management policies (e.g. anonymization, obfuscation) for ensuring privacy regulations and trigger alarm functionalities. This is done through the *LOCUS APIs* responsible for data acquisition and sharing within the components. The horizontal representation of these functions in the schema (Figure 3.1) emphasizes the need to apply these functions to all data and external as well as internal interfaces.
LOCUS APIs	The LOCUS APIs include all those interfaces that drive the exposure of the resulting location information coming from the Localization Enablers as input to the relevant system components: Localization & Analytics for Smart Network Management; and Localization & Analytics for New Services. They facilitate the storage of the related outputs in the LOCUS Persistence Entity for reuse by various components, regulating the access to data, while providing traditional access control functionalities for enabling authorization and authentication for the exchanged data within and outside the LOCUS system. The LOCUS APIs also include dedicated interfaces for exposing LOCUS localization analytics data as a service towards new services and smart network management applications.
LOCUS Management and Orchestration (MANO)	All the above-mentioned LOCUS functions are subject to automated lifecycle management through dedicated *LOCUS MANO* features. *LOCUS MANO* enables the automated instantiation, configuration, and operation of the various localization and analytics, which are deployed in the form of virtualized network functions (VNFs) on top of the virtualized infrastructure, following the ETSI NFV principles for functions virtualization and their management. The LOCUS MANO also enables the packaging of the LOCUS functions as virtualized functions

	for their share and re-use across different localization analytics services.
Virtualized infrastructure and Platform	The virtualized platform is implemented as a 5G-enabled edge and core virtualized infrastructure capable to offer isolated running environments for the various LOCUS functions. LOCUS makes use of de-facto standard virtualization infrastructure managers like Openstack [20] and containers orchestration tools like Kubernetes [21] to realize this.

Lastly, the localization analytics exposed by the LOCUS platform will feed the SNM and NSE/3rd party applications, therefore enabling various use cases related to network management and other verticals. These applications are not part of the LOCUS platform per se, but selective implementation of these applications related to the use cases described in D2.1 [1] are within the LOCUS project scope, offering an end-to-end view of the LOCUS platform capabilities and proof that the various applications can benefit greatly from the LOCUS platform analytics. In the same context, any aspects related to network control and enforcement of network actions, therefore to the appropriate APIs towards the OSS/BSS, are not part of the LOCUS platform scope, but of the SNM Applications per case.

4 Functional architecture

This section describes in detail the functional architecture of the LOCUS platform. Starting from the high-level architecture, described in the previous section, the main functions of the LOCUS platform are enumerated and presented following the generic approach described in Section 4.1. The various functions covered in this chapter refer to the data collection, localization analytics, ML Functions and related pipelines for enabling the LOCUS analytics.

4.1 Generic LOCUS function design

Here, a generic design approach is presented for describing the LOCUS localization, analytics, ML and security/privacy functions. These functions are part of the LOCUS platform and -when combined- they produce the localization analytics services to be exposed to the Smart Network Management and vertical/3rd party applications. As shown in Figure 4.1, each of these LOCUS functions is described as the combination and integration of different features and capabilities:

a) The data input, e.g. what type of data is expected, parameters like source of data, periodicity, protocols etc.
b) The purpose of the function itself, i.e. what does it do and what type of processing or ML/DL takes place, description of related parameters such as storage and computational needs etc.
c) The expected output/analytics services produced, that can be either stored in the LOCUS Persistence Entity or consumed by other LOCUS functions.

With respect to Figure 4.1, not all of the LOCUS functions are required to provide all of the features shown, for example ML capabilities may not be relevant in some cases.

Moreover, all the functions implemented follow this generic design and are subject to be managed and orchestrated on top of the virtualized infrastructure through the LOCUS MANO. The latter takes care of their on-demand instantiation and configuration (e.g. to set appropriate parameters for data processing and analytics exposure) according to the specific needs LOCUS Smart Network Management or vertical/third party application at a given moment. For this purpose, each LOCUS function (as depicted in Figure 4.1) includes dedicated management and control interfaces exposed to the LOCUS MANO for its automated configuration and operation. In addition to this, many of these function templates can be decomposed as a series of functions wired and chained together (Section 4.4), in order to realise the localization analytics exposed to the LOCUS application.

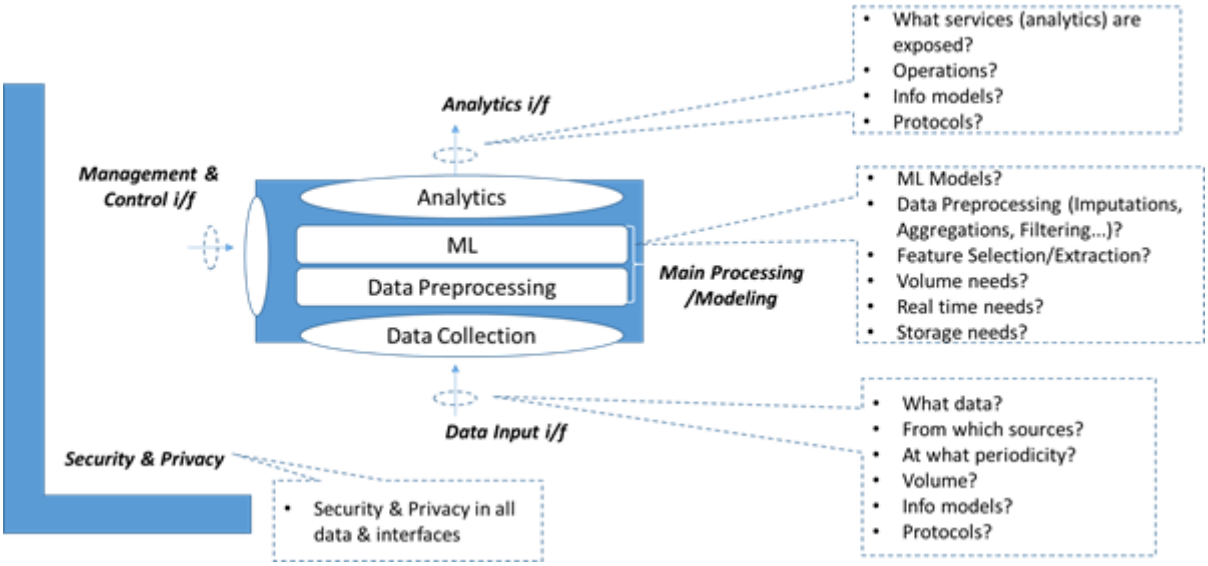


Figure 4.1 Generic Functional Block Design

4.2 LOCUS functions templates

4.2.1 Data Collection Functions

This LOCUS function is responsible for collecting the data from various sources outside the LOCUS platform so as to accommodate all the LOCUS proposed functionalities, therefore cater to the delivery of the appropriate "Localization Analytics as a Service" by the LOCUS platform.

4.2.1.1 High-level Requirements

A set of high-level requirements with respect to the related data collection functions can be derived based on the use cases described in detail in D2.1 [1], still subject to updates and future requirements to be considered as they refer to ongoing activities. While the definition of these functions will be finalized in the final version of this deliverable, the general categories of data expected to be employed in the LOCUS system are defined as:

- *Network data*, i.e. positioning-related measurements (e.g. Received Signal Strength-RSS, Time of arrival-ToA, Angle of arrival-AoA, Difference of arrival-DoA, Time difference of arrival-TDoA) acquired through measurement campaigns, *3GPP location data* through 5G Core Location Management Functions (LMFs) defined in the 3GPP TS 29.572 [22], *5G network Key Performance Indicator (KPI) data* (e.g. cell-level or area-wide network KPIs such as traffic coming from network monitoring and management analytics), and *Network configuration data* (i.e. topology, configuration parameter information).
- *User device data*, referring to additional positioning data and information from user devices, such as Global Positioning System (GPS), barometer and others.

- *Environment/External data, such as environment and other external auxiliary to position-related data***,** i.e. data from Internet of Things (IoT) devices (4G, WLAN or others), e.g. cameras, sensors, as well as auxiliary data (indoor maps, building boundaries and so on). *Social data,* can be considered a specific subcase of external data. They can be either directly collected from 3rd Party Apps or produced by LOCUS analytics components based on collected external information.

Another relevant data collection aspect refers to (a) the data rate in which the information is acquired and (b) if it is acquired through:

- A Polling process (Poll), i.e. a periodical checking for data to be fetched, or
- *A Publish-Subscribe (Pub-Sub) mechanism*, appropriate for streaming data; where the subscriber(s) express interest in specific information and only receive messages by a publisher that are relevant to the expressed interest.
- *An Asynchronous Request-Response (Req/Resp)*, where an interaction is initiated by sending a Request message and the system matches the Response message that is sent by the provider at some unknown later time.

Other data collection aspects include the specific technology adopted for accessing such data sources outside the platform. In LOCUS, two main approaches in terms of APIs are envisaged to collect data:

- HTTP REST services: in this case, the data collection function makes use of an HTTP REST API to directly query as client the external data source. As an alternative, the data collection function itself could offer an HTTP REST API (server side) to receive the data from the external source (e.g. in the case of pre-existing datasets to be loaded in the LOCUS platform)
- Message bus (e.g. Kafka [23]) services: this approach is used when the external data source is able either to expose and provide a message bus itself, or when the message bus is offered by LOCUS data collection function. In both cases, the LOCUS data collection function acts as a consumer of the continuous stream of data (while the external data source acts as a producer)

4.2.1.2 Data Collection Functions Template Definition

This section introduces the functions that are used by the LOCUS platform in order to access external information, either to be stored via the various Persistence functions, or to be provided/forwarded to a next processing layer (e.g. a real time service). The data collection functions are defined based on the following aspects; the Data Collection Function Target (i.e. the source of the data), the Data Collection Interaction Mechanism and the Data Type.

4.2.1.2.1 Data Collection Function Target

A data collection function will always involve a target entity to which the collection process will be addressed. Different sources implicitly define different transport protocols and parameterization for their interaction. To select a function target for a data collection template we must also specify the parametrization required accordingly. Table 4.1 enumerates the various targets and parameters.

Table 4.1 Data Collection Targets & parameters

Function Target	Description	Parameters
Internet Sources	Various http-based (e.g. REST API) or other internet-friendly protocols	URL per endpoint, authentication-specific parameters
Google or IOS specific cloud services	Cloud services integrated with mobile phones	User Access tokens, type of service accessed (e.g. google firebase or google account analytics)
3GPP Network sources	Network-related operational information (network layers for UE or RAT devices, system aspects, network meta-data) provided via direct or indirect interaction with the NME	3GPP compliant description of the wanted resources via JSON[1], XML[2], YAML[3], ASN.1[4], graphQL[5] or other language
3GPP non-Network sources	Non-Network data provided via direct or indirect interaction with other 3GPP-specific entities (e.g. LMF, 5G localization entities, Smart City entities)	3GPP compliant description of the wanted resources via JSON[1], XML[2], YAML[3], ASN.1[4], graphQL[5] or other language
Non-3GPP network sources	Other Sources of network data that can be accessed for accessing non 3GPP related information, e.g. SNMP (Simple Network Management Protocol), OpenFlow enabled routers/cloud switches)	Target Controller IP/Port and connection configuration. Definition of the wanted resource (SNMP Object Identifier-OID or OpenFlow field identifier)

[1] JavaScript Object Notation, https://www.json.org/json-en.html
[2] (e)Xtensible Markup Language https://www.w3.org/TR/xml/
[3] YAML Ain't Markup Language https://yaml.org/
[4] Abstract Syntax Notation https://www.itu.int/en/ITU-
[5] Graph Query Language https://spec.graphql.org/June2018/

4.2.1.2.2 Data Collection Interaction Mechanism

The interaction mechanism refers to the way that the data collection operation will be performed. In general, this can be split into two categories, active (system-initiated data collection) and passive (environment-initiated data collection).

- *Passive (environment-initiated)*
  - LOCUS external REST API; this way new data is pushed towards our system via POST/UPDATE http methods or another http implementation.
  - LOCUS message bus; e.g. a Kafka consumer that waits for messages from an external Kafka queue.
- *Active (LOCUS-initiated, automated or manual)*
  - Initial/one-time extraction; this is the general case for acquiring a large dataset as a non-repetitive task (e.g. extract an open dataset).
  - Assume New State; i.e. every time a resource is accessed, a new measurement is assumed and acquired. For this case, a necessary input parameter would be the polling interval of the data collection function.
  - State Difference Recognition; i.e. access to a resource for which no notion of state is available and perform updates, incremental or not on a local model. Additional input parameter that is required is the implementation of the comparison process (identification of the new state).
- *Combined approach*
  - In some cases of data collection interactions, a hybrid approach for the integration may be necessary. For example, a public REST API could be used to receive notifications for new data being available in a public server (including the file path and other configuration) and then the active mechanism will be activated in order to acquire the dataset and ingest it into the platform.

4.2.1.2.3 Data Collection Function Data Type

The Data Types of the data collection functions are presented below. These categories are then linked to the available persistence storage options (different collections), as their respective query engines will assist on the data manipulation process with the best technology available.

- *System entity meta-data*: System entities can vary, including different categories such as network elements, user equipment (UE) devices, customers, cars, i.e. single identifiable entities that are involved into the LOCUS functional architecture. Their meta-data are static, non-temporally correlated information of various data types (e.g. cellname, sitename, radio configuration parameters).

- *System entity time-related data*: For the aforementioned System entities, data that are correlated with time (single timestamp or duration) are denoted as 'entity time-related' data and can vary from different contextual fields (location, network KPIs, sensor input data etc.).
- *Geometry data* (e.g. areas, maps): geometry meta-data, i.e. data representation of spatial structures and definition of spaces (e.g. KML[6], XML, Geo-JSON[7]).
- *MultiMedia data & metadata (context and time)*: Multimedia must be specially handled by the LOCUS platform and they usually refer to image, sound, video data or other special cases (e.g. spectrum).

In the following Table (Table 4.2), the various data, derived from the Functional Requirements of the LOCUS platform and presented in the deliverable D2.1 [1], are mapped in the aforementioned Data Targets, Interaction Mechanisms and Data Types.

Table 4.2 Mapping of data types derived from the functional requirement analysis to specific Data Targets, Interaction Mechanisms and Data Types

Data Description	Data collection Data Target	Data collection Interaction mode	Data Collection Function Data Type
Position-related measurements (e.g. RSS, ToA, AoA, DoA, TDoA)	3GPP network Sources	Active	System entity time-related data
User device location information in the 5G network through 5G Core Location Management Functions (LMFs), as defined in the 3GPP TS 29.572 [22]	3GPP non-Network sources	Active	System entity time-related-data
5G network KPI data, including cell-level or area-wide network KPIs (e.g. traffic, flows, etc.) coming from network monitoring and management	3GPP network Sources	Active	-System entity time-related data

[6] Keyhole Markup Language https://www.ogc.org/standards/kml
[7] Geospatial Data JSON https://tools.ietf.org/html/rfc7946

analytics, if required by interacting with 5G Core network functions			
Network topology and configuration parameters information related to network devices	3GPP network Sources	Active – One-time	System entity meta-data, geometry data
Additional positioning data and information from user devices, such as GPS, barometer and others	-Internet Sources -Google or IOS specific cloud services	Active	System entity time-related data
Environment data, i.e. system communicate directly (online) with various IoT devices (e.g. cameras, sensors)	-Internet Sources -Non-3GPP network sources (e.g. IoT gateways, camera gateways, media servers)	Passive	System entity time-related-data, Multimedia & metadata
Auxiliary data and specifically access to open data cloud services, and metadata like indoor maps, building boundaries and so on.	-Internet Sources	Active - One-time	System entity meta-data, geometry data, Multimedia & metadata
Social data	-Internet Sources -Google or IOS specific cloud services	Active	-System entity meta-data, System entity time-related data
Non-3GPP positioning-related measurements e.g. WiFi	-Non-3GPP network sources (e.g. Cisco CMX)	Active	-System entity time-related data

4.2.1.2.4 Data Collection Function Template

Based on the data collection requirements and characteristics defined above, the Data Collection Function Template can be defined as follows in the table below.

Table 4.3 Function: Data Collection

Function: Data Collection

Input:
• Data Collection Function Target (one of the available targets enumerated)
  ○ Function target sub-parameters (e.g. for internet REST API specify the URL)
• Data Collection Function Interaction Mechanism (one of the available interaction mechanisms enumerated)
  ○ Function Interaction mechanism specific parameters (e.g. for Active select the sampling interval)
• Data Collection Function Data Type (one of the available high-level function data types enumerated)
• Callback for the invocation of other LOCUS platform function
• Callback behaviour based on the various data collection status signals
• Failure handling

Description: This function's purpose is to collect the various data types from external -to the LOCUS platform- sources with the selected interaction mechanisms (as described in the previous sections).
Output: • Batches or Slices of data from the various data sources defined by the Data Collection Function Target and parametrization • Signalling data for the data collection operation (logs included)

4.2.2 Localization Enablers

The Localization enablers provide the localization mechanisms by utilizing the properties of radio signals associated with a given UE (or the user). These enablers studied in the LOCUS project are divided into 3 categories. These are the following:

- *3GPP Technologies:* The localization of a UE is performed using the collected radio signals that have been transmitted from 5G NR, LTE, UMTS or GSM antennas.
- *Non-3GPP Technologies:* The localization of a UE is performed using the collected radio signals that have been transmitted from e.g. WiFi Access Points etc.
- *Device-free Technologies:* The localization of a UE/object is performed using a passive monitoring system that can locate the object position without the subject's participation, where the subject does not need to carry any radio device.

In all of the above types of localization, the user consent for this in the privacy settings is taken into account, except for emergency and regulatory cases (where the laws allow for). A dedicated section related to location data privacy and security follows.

The following tables define the three main localization enabler functions templates.

Table 4.4 Function: 5G-based LEN

Function: 5G-based LEN
Input: • 3GPP defined radio measurements to compute time delay of arrival (Positioning Reference Signal-PRS and Sounding Reference Signal-SRS), angle of arrival/departure (Single-Sideband modulation-SSB components), received signal strength (Reference Signal Received Power-RSRP, Reference Signal Received Quality-RSRQ, Received Signal Strength Indicator-RSSI) and cell ID indications (CID and e-CID). • Locations of fixed and/or ad-hoc access points (these through GPS or GNSS) • Physical and radio geometrical data of the area of interest. • Accuracy, reliability and update rate criteria
Description: This function's purpose is to localize any type of 5G mobile device, by leveraging multi-RAT, multi-carrier and mmWave technologies.
Output: • The derived location of the device, which should meet the accuracy, reliability and update rate criteria as set by the service KPIs

Table 4.5 Function: Hybrid (5G & non-3GPP) LEN

Function: Hybrid (5G & non-3GPP) LEN
Input: • 3GPP based localization inputs as noted above. • Non-3GPP localization inputs related to received signal power, time delay, angle of arrival/departure and access point IDs. • Area of interest geometrical data (definition of physical sections, streets, areas and definition of radio coverage geometry, location of 3GPP and non-3GPP access points. and coverage zones/polylines) • Accuracy, reliability and update rate criteria
Description: This function's purpose is to fuse 5G network data and data/information derived from non-3GPP technologies with the explicit purpose of enabling device localization.
Output: • The derived location of the device, which should meet the accuracy, reliability and update rate criteria as set by the service KPIs

Table 4.6 Function: Device-free LEN

Function: Device-free LEN
Input: • Cellular and/or non-cellular signals for localization of device-free targets. • Locations of the cellular and/or non-cellular signal emitters. • Accuracy, reliability and update rate criteria
Description: This function's purpose is to enable the detection and tracking for single and multiple device-free targets.
Output: • The derived location of the target, which should meet the accuracy, reliability and update rate criteria as set by the service KPIs

4.2.3 Analytics Functions

LOCUS can provide various analytics that leverage geolocation data, either in real-time or non-real-time. These analytics functions can be integrated and chained within wider scope localization analytics services (e.g. including cascades of analytics and ML functions), or be consumed and directly exposed to Smart Network Management and vertical/3rd party applications, so as to enable use cases related to the following categories:

- People Mobility & Flow Monitoring.
- Network-Assisted Self-Driving Objects.
- Smart Network Management based on Localization.

Coupled with the ability of the LOCUS platform to deploy ML/DL models and expose the Localization Analytics as a Service for the Application layer, these functions enable advanced analytics capabilities leveraging position information and encompass the following:

- *Descriptive Analytics*: Perform statistical analysis on collected data (What happened?).
- *Predictive Analytics*: Assess future possibilities, given collected data (What can happen?).
- *Prescriptive Analytics*: Search for actions to be taken, given data (What to do to make it happen?).
- *Diagnostic Analytics*: Determine the causes for specific outcomes from data (Why did it happen?).

The following tables present the Statistics/ML/DL-based functions in terms of data requirements and main functionalities provided.

Table 4.7 Function: Trajectories Identification

Function: Trajectories Identification
Input: • Dataset of historical geolocation data as generated from localization functions • Area coordinates under investigation • Time window in which trajectories will be extracted
Description: This function refers to the identification of mobility trajectories extracted from specific areas of interests, and within a specific time window of interest. This functions output is agnostic to the localization source, identified trajectories could come from the positioning source considered the most accurate at the moment of execution, or could come from the fusion of different sources information. This function could trigger a predictive model to forecast trajectories in case the request time window includes future timesteps.
Output: • List of identified distinct trajectories present in the designated area and the designated time window

Table 4.8 Function: POI Identification

Function: POI Identification
Input: • Dataset of historical geolocation data as generated from localization functions. • Area coordinates under investigation
Description: This function refers to the identification of geometric areas of interest (Points of Interest - POIs) that result from high population density and concentration. In general, we expect that marketplaces, shopping malls, popular streets and venues will be identified by this function.
Output: • Geometric Representation of the POIs persisted in the LOCUS persistence context

Table 4.9 Function: Routes identification

Function: Routes identification
Input: • Dataset of historical geolocation data as generated from localization functions • Coordinates of Point of Origin

• Coordinates of Point of Destination • (Optional) Coordinates of Intermediate Points • Time window/constraint
Description: This function refers to the identification of one or more possible path(s) that lead from a Point of Origin to a Point of Destination under the time constraint requested. In general, we expect that this function identifies the best route, but in case of multiple paths are possible, multiple options will be provided. In case no option is possible, no path will be provided, instead, estimation of the minimum time constraints could be provided. This function can trigger a predictive model in case the request time is in the future.
Output: • List of identified routes (timed trajectories) that LOCUS estimates will lead from Point of Origin to Point of Destination in the specified time window • An estimation of minimum time window, in case the request cannot be satisfied

Table 4.10 Function: Flows identification

Function: Flows identification
Input: • Dataset of historical geolocation data as generated from localization functions • Coordinates of areas under investigation • Flow tiles configuration • Time window under investigation
Description: This function refers to the identification of the Origin Destination Matrix, of velocity fields/vectors over the designated area, computed within the designated time window. This function can trigger a predictive model in case the request time spans future timesteps.
Output: • An Origin Destination Matrix of the mobility flow (e.g. as represented by velocity fields)

Table 4.11 Function: Counts identification

Function: Counts identification
Input: • Dataset of historical geolocation data as generated from localization functions • Coordinates of areas under investigation • Measurement Time

Description: This function refers to the identification of how many separate individuals are present in a designated area at a specific time. By means of sensor radar networks, counting can also be performed without association with position data (device-free approach).This function can trigger a predictive model in case the requested time is in the future.
Output: • A count of individuals identified in the designated area at the requested time

Table 4.12 Function: Feature Changes Tracking

Function: Feature Changes Tracking
Input: • Continuous stream of historical geolocation data as generated from localization functions • Coordinates of areas under investigation • Feature of interest, with a specific change condition to be tracked • Optional Time window
Description: This function refers to a subscription request to track a designated feature (velocity, position, acceleration, direction, proximity, grouping, etc.) of an individual/trajectory or a group/trajectory in a designated area within (optionally) a designated time window.
Output: • A notification with a list of elements (groups, individuals, trajectories...) when the requested condition is satisfied

Table 4.13 Function: Time to Collision

Function: Time to Collision
Input: • Continuous stream of historical geolocation data as generated from localization functions • Coordinates of areas under investigation
Description: Time-to-Collision (TTC) estimates the likelihood of a collision of moving objects. See TTC calculation example for Vulnerable road user (VRU) in [24], section 6.5.10.5.
Output: • Estimation of Time to Collision and Likelihood of Collision

Table 4.14 Function: Profiles Identification

Function: Profiles Identification
Input: • Dataset of historical geolocation data as generated from localization functions • Coordinates of areas under investigation • Optional list of user/trajectory IDs
Description: Within the designated area, this function identifies the provided trajectory/user's mobility profile, from a set of possible profiles. If no IDs are requested, identify the profiles of all available trajectories in the designated area.
Output: • A mapping between Trajectory/User IDs and their profiles

Table 4.15 Function: KPI heatmaps

Function: KPI heatmaps
Input: • Dataset of historical geolocation data as generated from localization functions • A list of KPIs of interest, with their parameters (geographic scope, time window, etc.)
Description: This function refers, to the generation of a heatmap view, from the historical geolocation data. This function might trigger a predictive model to generate information if the requested time window spans future timesteps. The KPIs may refer to historical and/or near future KPIs, such as network-related measurements (e.g. Signal to interference and noise ratio - SINR) and may refer to selected subareas that can be ML-derived, such as mapping KPIs in detected trajectories.
Output: • Visualization ready, list of heatmap data structures

Table 4.16 Function: Contextualized Indicator Generation

Function: Contextualized Indicator Generation
Input: • Dataset of structured historical data of interest, that include UEs' time-series position and network measurement data • Base station positions

• (Optional) Coverage maps or propagation models results • Information on the areas of interest to be considered. Optionally this function may rely on an automatic generation of areas of interest • Parameters related to geographic scope and time window under consideration
Description: This function refers to the fusion of diverse time-dependent data in a selected area and time-frame from the historical geolocation data in order to create new indicators of time-series format as a result of this data fusion. These new indicators can be directly used as input for inference mechanisms (i.e. fed to other analytics components referring to different functional (sub)blocks as an alternative to full data input) and improve their results in comparison to employing purely network measurement-based inputs. To reduce the generated possible huge amount of indicators (due to the huge amount of possible areas of interest to be considered) ML algorithms for dimensionality reduction and feature selection can be applied after the generation of the indicators.
Output: • New contextualized indicator values

Table 4.17 Function: KPI Prediction

Function: KPI Prediction
Input: • Targeted network KPI(s) to be predicted • Set of data to be included in the prediction process, including network, position/mobility and other external data • Modelling (Analytics/ML/DL) approach and related parameters employed • Area coordinates and time-frame under investigation
Description: This function uses ML and/or DL methods to predict network-related KPIs, such as network demand and Quality of Experience metrics, in the near future.
Output: • Time-stamped predicted values for selected KPI(s)

Additional LOCUS basic analytics functions, based mostly on metric/location data manipulation and statistics are described in the following tables.

Table 4.18 Function: Geometric Area Correlation

Function: Geometric Area Correlation
Input: • A set of geometrical areas to be correlated • Parametrization of the correlation process (incl. correlation method)
Description: This function refers to the computation of correlation between a set of selected input geometrical areas.
Output: • Correlation results (value per distinct area pair)

Table 4.19 Function: Data Filtering

Function: Data Filtering
Input: • Multidimensional dataset (structured or unstructured) • Filtering criteria (metric, time, or entity context) • Desired columns or sections of the original dataset
Description: This function refers to the transformation of a dataset to a subset of itself by applying computational logic.
Output: • New multidimensional dataset (filtered) based on the filtering criteria and the selected columns

Table 4.20 Function: Metric aggregation

Function: Metric aggregation
Input: • Multidimensional Dataset (Structured) • Aggregation Criteria (selected KPIs, Aggregation Columns, Metric Aggregation Functions – average, sum, maximum etc.)
Description: This function refers to the transformation of a dataset into a new dataset by applying the aggregation function and reducing its dimension to the selected aggregation parametrization.

Output:
• New dataset (with aggregated metric values) based on the aggregation criteria

Table 4.21 Function: Categorical Aggregation

Function: Categorical Aggregation
Input: • Multidimensional Dataset (Structured) • Aggregation Criteria (Selected Categorical Features, Aggregation Columns, Categorical Aggregation Functions – Majority, Distribution etc.)
Description: This function refers to the transformation of a dataset into a new dataset by applying a categorical-specific aggregation function that will reduce its dimension to the selected aggregation parametrization.
Output: • New dataset (with aggregated categorical values) based on the aggregation criteria

Table 4.22 Function: Structured Dataset Joining

Function: Structured Dataset Joining
Input: • More than 1, multi-dimensional datasets • Criteria to be used for the joining operation (e.g. join on column1 = column2)
Description: This function refers to the transformation of a set of structured multi-dimensional datasets into a single, joined dataset, by the means of a structured joining operation.
Output: • The new generated joined structured dataset

Table 4.23 Function: Geo-query

Function: Geo-query
Input: • List of geometric data points (lat, lng) • List of geometric areas

Description: This function refers to the process of identifying the data points (lat, lng) pairs that belong to or are relevant to a specific geometric area.
Output: • Report of correlation between the points and the areas

Table 4.24 Function: Reverse Geocoding

Function: Reverse Geocoding
Input: • Input geolocation data (single, multi data points) • Geocoding specification (contains, adjacency parameters)
Description: This function refers to the transformation of input geolocation data into geometric areas that are correlated or contain these location data.
Output: • List of Geometric Areas that are correlated with this location • (Optional) Adjacency, relevance, accuracy

Table 4.25 Function: Geometric Area Manipulation

Function: Geometric Area Manipulation
Input: • Geometric areas that will be involved in the manipulation operation • Type of operation (intersection, union, split)
Description: This function refers to the transformation of new geometric areas based on other geometric areas, manipulation operations and parametrization. The result will be a new, edited geometric area.
Output: • New geometric areas generated by the manipulation operation

4.2.4 Security & Privacy Functions

This subsection presents the LOCUS architecture functions related to the security and privacy aspect. As presented in Figure 3.1, the LOCUS architecture includes a functional block specifically dedicated to location security and privacy functions. In the designed architecture,

security and privacy requirements have strong dependencies with other LOCUS functions. Location security and privacy functions are applied to different parts of the LOCUS architecture through the specific functions interface, enabled by the LOCUS APIs. The main properties required for location base service in 5G networks can be summarized as follows:

- Security Requirements
    - **Authentication**: is one of the building blocks of security measures and it protects private and confidential user information. Authentication is the process of identifying entities (i.e. users) accurately.
    - **Confidentiality**: is necessary to protect messages contents from passive eavesdroppers. Confidentiality guarantees the delivery of messages to designated recipients or authorized parties.
    - **Integrity**: Integrity can ensure that the data cannot be tampered by intended or un-intended interference during the data delivery in communication networks, ultimately providing the accurate data for authorized users.
- Privacy Requirements
    - **Location Privacy**: the exact location information of the user must be protected from unauthorized entities. The user's trajectory which contains location data of the user's present and past locations including near POIs must not be revealed to unauthorized entities.
    - **Unlinkability**: is the process related to the user tracking protection that must be implemented to protect the users from linkability of two or more successive positions. In mobile networks, the Service Provider should be unable to link two or more successive positions of the user.
    - **Anonymity**: this means that the subject may perform an action without disclosing its user identity to third-parties.

The main goal for the LOCUS platform definition is to separate security and privacy requirements from the location-based application modules. This allows the LOCUS platform modules to focus on specific localization features using a common approach to retrieve security and privacy functions. In order to achieve this goal, LOCUS supports all the security requirements related to mobile devices, network, and third-party entities. In this context, all transactions between these three actors go through the functions interface. The interface is in fact acting as a connection between the Security & Privacy Layer, and the appropriate APIs exposing functionalities, and 3rd party applications for network management and other vertical applications that shall exploit localization analytics as a service by the LOCUS platform. The main advantage of this approach is that the security and privacy functions have the ability to monitor and modify all the content, maintain different policies for the device and evaluate possible attacks. The Location Security & Privacy Functions shall ensure all data privacy and

security aspects through the LOCUS APIs which handle data acquisition and sharing among the components.

4.2.4.1 Security functions

The following tables detail the security functions supported and provided by the LOCUS platform.

Table 4.26 Function: Authentication

Function: Authentication
Input: • User profile metadata (e.g. username, password hash) • External metadata related to venue/area maps and buildings/shops • Authentication type
Description: This function performs the user/device authentication and authorization process. It implements advanced cryptographic techniques for all network interfaces (homomorphic encryption, secure multiparty computation, and secure conditional sharing techniques), it will be in line with Lawful Interception Architecture defined in 3GPP [25].
Output: • User privilege and permission • User defined policy

Table 4.27 Function: Security Data Clustering

Function: Security Data Clustering

Input:
• Multi-dimensional numerical dataset
  ○ Location measurements provided by LOCUS location enablers
  ○ Location measurements provided by 5G network
  ○ I/Q samples by 5G network (raw data)
  ○ Data from other sensors
• Selected clustering algorithm
• Selected hyperparameters with respect to the Selected Algorithm

Description: This function implements ML models for Security purposes (e.g. Anomaly detection). Specifically, it uses clustering algorithms for anomaly detection. It works in the original domain or in a transformed domain of the received waveform (Wavelet Transform, Wigner-Ville transform, etc.). The latter may lead to an increase of the separability between the useful signal component features/parameters and those of the interference signal. The candidate algorithms shall process data in order to detect possible anomalies by clustering

data in either the original or a transformed domain and to somehow mitigate the erroneous measurements.
Output: • Ongoing attack flag • Measurement labels • Mitigated measurements

Table 4.28 Function: Security Data Cleaning

Function: Security Data Cleaning

Input:
• Multi-dimensional numerical dataset
  ○ Location measurements provided by LOCUS location enablers
  ○ Location measurements provided by 5G network
  ○ I/Q samples by 5G network (raw data)
  ○ Data from other sensors
• Selected clustering algorithm
• Selected hyperparameters with respect to the Selected Algorithm

Description: The LOCUS platform needs to ensure that the provided location estimates are reliable. This function monitoring estimation degradation can be effective to detect the presence of a malicious attack. Specifically, the goal of this function is to use summary statistics (e.g. mean, standard deviation, kurtosis, skewness) to feed algorithms (e.g. Kalman filters and particle filters) and detect malicious attacks. This function is aimed at the detection and (possibly) mitigation of malicious attacks. Specifically, the algorithms contained in this function shall face with both (noise) jamming and spoofing/meaconing attacks. The exploited techniques shall rely on the tools provided by statistical signal processing ranging from compressed sensing approach, tracking algorithms up to machine learning strategies.
Output: • Ongoing Attack Flag • Measurement labels • Mitigated Measurements

4.2.4.2 Privacy functions

The following tables detail the privacy functions supported and provided by the LOCUS platform.

Table 4.29 Function: Sanitization

Function: Sanitization
Input: • Continuous stream of historical geolocation data as generated from localization functions
Description: This function implements the process of removing user sensitive information from stored location data, so that the data may be distributed to a 3rd party entity. Sanitization attempts to reduce the data classification level. The goal of this function is also related to data anonymization. According to the user/device policy, the function can also apply encrypting on location data.
Output: • Continuous stream of geolocation data as it results from the pre-processing function

Table 4.30 Function: k-anonymity

Function: k-anonymity
Input: • Dataset of historical geolocation data, also sanitized (e.g. from LOCUS persistence entity) • Coordinates of areas under investigation
Description: This function implements data process to produce that any individual entity in the database is indistinguishable, with respect to the quasi-identifiers, from k - 1 other individuals. There are various methods to achieve k-anonymity: generalization of an attribute (for example postal addresses can be generalized to the street or to the city, depending on how much we need to generalize), suppression of an attribute, or addition of dummy records.
Output: • Dataset of historical geolocation data as it results from the pre-processing function

Table 4.31 Function: Obfuscation

Function: Obfuscation
Input: • Dataset of historical geolocation data, also sanitized (e.g. from LOCUS persistence entity) • Coordinates of areas under investigation

Description: This function implements techniques that aim at blurring or perturbing the location information contained in LOCUS persistence entity because of its potential sensitivity. As an example, the precision of location information can be decreased by translating precise point coordinates to geographic regions; Analogously, the precision of the temporal information usually associated with location can be decreased by converting precise timestamps into time intervals.
Output: • Dataset of historical geolocation data as it results from the pre-processing function

Table 4.32 Function: Policy definition

Function: Policy definition
Input: • Set list of restriction specifications that list location data fields that must have K unique values in the result set and a percent level of obfuscation • Set list of operation restriction specifications that deny access to a list of user location data fields and attributes except via the list of utility functions
Description: In order to manage different levels of privacy and security, the LOCUS platform provides the possibility to define user/device policy. This function implements the setting of the privacy policy. It describes a fine-grained sanitization policy developed for private data, to facilitate their release to public, and a sanitization tool that applies the policy. This policy works in the following way: i) query result restriction specifications that list fields that must have K unique values in the result set; ii) operation restriction specifications that deny access to a list of fields and variables except via the list of utility functions. This function has implications on filtering, aggregation, anonymization, and obfuscation procedures.
Output: • Policy definition result

Table 4.33 Function: Result Aggregation

Function: Result Aggregation (query)
Input: • Query description to be applied on the unstructured dataset's index • Persistence Entity context
Description:

LOCUS platform protects data location through a secure query framework which only releases aggregate and prevalent results based on work done by [26] and [27]. Thus, the privacy of a user/device is further protected by using a "hiding in a crowd" approach 3rd party can query on any features which interest them but they only receive aggregate responses (counts, histograms, etc.) to address data query correlations. These responses are synthesized from user/device individual responses, after applying their query policies to ensure.
Output: • Structured Dataset as it resulted from the query operation

4.2.5 Machine Learning Functions & Pipelines

In the following subsections, the elementary ML/DL-related functions are presented (Section 4.2.5.1). These functions -appropriately parametrized- will be chained in ML pipelines (Section 4.2.5.2) so as to enable localization analytics services to be created on-demand and subsequently exposed through the LOCUS APIs.

4.2.5.1 ML Functions

The elementary ML/DL-related functionalities offered by the LOCUS platform are presented in this section. The following tables provide details for these LOCUS functions, in terms of data requirements and main features provided in the context of the LOCUS UCs.

Table 4.34 Function: Data cleaning

Function: Data cleaning
Input: • Multi-dimensional raw dataset (structured or semi-structured) • Data cleaning specification that includes the order of operations and parameters of each operation
Description: This function refers to the data cleaning pipeline that will take as input a raw, unprocessed data as found in the persistence context of the LOCUS platform. The function will include several data cleaning operations such as impute/interpolate/predict the missing data values, smoothen the data using regression or other methods, remove outliers using simple clustering or other methods, create attributes from the existing attributes, if required.
Output: • Multi-dimensional numerical structured dataset as it results from the data cleaning function

Table 4.35 Function: Dataset Pre-processing

Function: Dataset Pre-processing
Input: • Multi-dimensional raw dataset (structured or semi-structured) • Preprocessing pipeline specification including order of operations and parametrization per operation
Description: This function refers to the processing pipeline that will take as input a raw, unprocessed dataset as found in the persistence context of the LOCUS platform and prepare it to be used into various analytical and ML functions that require numerical representations. This function can include several operations like data transformation, data encoding to other formats like binary, ordinal or one-hot, categorical or label-encoding, feature scaling (data standardization and normalization), creating training and test splits for a supervised learning.
Output: • Multi-dimensional numerical dataset as it results from the preprocessing function

Table 4.36 Function: Multiple Regression

Function: Multiple Regression
Input: • Multi-dimensional numerical dataset • Selected hyperparameters (linear/polynomial, degree, etc.)
Description: This function refers to training a regression model based on two or more independent variables (or regressors or predictor variables) to predict a dependent forecast variable.
Output: • A regression model from the training • Predictions/forecasts from the regression model • (Optional) Performance evaluation of the training

Table 4.37 Function: Multivariate or Spatial Interpolation

Function: Multivariate or Spatial Interpolation
Input: • Multi-dimensional numerical dataset • Selected method and parameters

Description: This function refers to a task to estimate the variables at unobserved geo-spatial locations based on the data from the observed locations. Existing methods like inverse weighted distance (IDW), kriging, nearest neighbours (NN), thin plate spline (TPS), radial basis function (RBF), local polynomial interpolation (LPI) can be used for spatial interpolation.
Output: • Spatially interpolated values for the missing input locations/observations

Table 4.38 Function: ML-based Dimensionality Reduction

Function: ML-based Dimensionality Reduction
Input: • Multi-dimensional numerical dataset to be reduced by the vector quantization method • Selected algorithm/method • Selected hyperparameters with respect to the algorithm
Description: This function refers to the transformation of a multi-dimensional numerical dataset into a reduced representation of itself by the means of dimensionality reduction schemes. This function will contain both the training and the reduction phase. This function can include existing ML-based dimensionality reduction techniques such as Principal Component Analysis (PCA), random forest, t-Distributed Stochastic Neighbour Embedding (t-SNE), uniform manifold approximation and projection (UMAP).
Output: • Multi-dimensional numerical dataset that is the quantized representation of the input dataset

Table 4.39 Function: Feature Ranking and Selection

Function: Feature Ranking and Selection
Input: • Multi-dimensional numerical datasets (features) for feature selection • Selected algorithm/technique • Selected hyperparameters for a technique
Description: This function refers to the process of selecting highly-informative variables (features) that are useful to build an efficient predictive model. This function can include ML algorithms

based on random forests like Boruta and/or RRF variable importance; LASSO regression, recursive feature elimination, stepwise forward and backward selection, simulated annealing etc.
Output: • Multi-dimensional numerical dataset that is the filtered and reduced from the input dataset

Table 4.40 Function: Hyperparameter tuning of a ML task

Function: Hyperparameter tuning of a ML task
Input: • Reference to specific instantiation of ML Task • Hyperparameter Space related to the selected ML Task • Methodology of tuning and hyperparameters of the search operation
Description: This function refers to a methodology in which an internal ML task is being repetitively executed with a goal to either minimize or maximize a specific ML KPI (e.g. minimization of mean square error)
Output: • Optimum Execution Result (ML Model) for the specific input hyperparameter space and referenced ML task

Table 4.41 Function: Classifier - Training

Function: Classifier - Training
Input: • Multi-Dimensional Dataset • Selected Classification Algorithm • Selected hyperparameters based on the Selected Algorithm
Description: This function refers to the training process of a ML model that will be used as a classification model for an input dataset. Exiting ML techniques like Support Vector Machine (SVM), k-nearest neighbours, decision trees can be used for this function.
Output: • ML model as it resulted from the training function • (Optional) Performance evaluation of the training function

• (Optional) Multi-dimensional dataset including the predicted classes of the classification

Table 4.42 Function: Classifier - Inference

Function: Classifier - Inference
Input: • Classifier model that will be used for the predictions • Multi-dimensional dataset that fits the specifications of the classifier model • (Optional) Additional multi-dimensional dataset that will be used for validation purpose
Description: This function refers to the prediction process of a classification ML Model. It takes as input a multi-dimensional numerical dataset that fits its input specifications and executes the prediction step, generating predicted labels for each data point.
Output: • Multi-dimensional dataset with the predicted class(es) per data point based on the selected model • (Optional) Evaluation of error of the KPIs based on the validation dataset

Table 4.43 Function: Clustering Model

Function: Clustering Model – Training and Inference
Input: • Multi-Dimensional numerical dataset • Selected clustering algorithm • Selected hyperparameters with respect to the selected algorithm
Description: This function refers to the execution of ML training and predict (merged) task that will generate information for underlying data clusters (based on the input dataset) and it will also produce a per-data-point cluster classification. The existing approaches like K-means clustering, k-medoids clustering, k-paths large scale trajectory clustering and/or density-based clustering (DBSCAN, HDBSCAN) will be used for this function.
Output: • Description of the identified clusters in the form of meta-data (e.g. centroids) • Per data point classification with a cluster from the result clusters • (Optional) Performance evaluation of the clustering execution

Table 4.44 Function: Space-time trajectory clustering

Function: Space-time trajectory clustering
Input: • Multi-dimensional numerical dataset • Selected hyperparameters
Description: This function refers to the execution of a ML algorithm to identify clusters in spatio-temporal datasets using agglomerative hierarchical clustering. Traditional clustering based on statistical techniques often fail when applied to moving objects and big data. This function based on space-time hierarchical clustering incorporates location, time, and attribute information to identify the groups across a nested structure reflective of a hierarchical interpretation of scale.
Output: • Identified clusters or hotspots and shape of the clusters that indicate the movement patterns for tracking • (Optional) Performance evaluation of the clustering using metrics like homogeneity score and average random index score

Table 4.45 Function: Time series predictions/forecasting

Function: Time series predictions/forecasting
Input: • Time-series input data, intervals (if any used), and target variable for prediction • Constituent components of time-series: level, trend, seasonality, noise (if the components are available) • Selected ML/DL approach and related hyperparameters
Description: This function refers to the training of a ML/DL model for time series forecasting/predictions. This function can include baseline methods like simple naïve or persistence forecasting and averaging methods, autoregressive forecasting method (ARIMA), or DL methods that using LSTMs, CNNs, or hybrids of CNN and LSTM models such as CNN-LSTMs, ConvLSTMs.
Output: • Predictions for the target variables • Trained ML/DL models from this approach • Optional performance evaluation of the ML/DL model

Table 4.46 Function: Static or mobile group detection

Function: Static or mobile group detection
Input: • Multidimensional numerical data set that contains geolocations generated from localization functions, coordinates of areas under investigation, time window under investigation • Selected approach and related hyperparameters
Description: This function refers to the execution of a ML-based method to detect crowd groups in short-time intervals and long-time linkages using multimodal data (like bluetooth, wifi etc.) in an indoor or outdoor environment. This function includes graph-based clustering algorithms such as density-based scanning approach on graph models (DenGraph), clustering highly connected subgraphs (HCS) or fully connected subgraphs (MaxClique).
Output: • Description of the identified static or mobile groups and group size • (Optional) Performance evaluation of the selected approach

Table 4.47 Function: Classifying moving object trajectories using representation learning

Function: Classifying moving object trajectories using representation learning
Input: • Multidimensional numerical data set that contains geolocations generated from localization functions with time stamps (trajectories of moving objects) • Annotations (labels) for the mode of transportation • Model hyperparameters
Description: This function refers to a DL-based method to estimate users' transportation modes from their movement trajectories. The steps involved in this method: generating informative trajectory images from raw trajectories, feature extraction from trajectory images using a fully-connected deep neural network with stacked denoising autoencoder, train a classifier (SVM, decision tree, or logistic regression) using the extracted features and annotations, and estimating the transportation modes using this classifier.
Output: • Estimates of the transportation mode • Trained ML/DL models from this approach

Table 4.48 Function: Deep probabilistic clustering with self-organizing maps

Function: Deep probabilistic clustering with self-organizing maps
Input: • Multidimensional numerical dataset (preferably time series data/trajectories) • Model hyperparameters
Description: This function refers to a DL-based method that combines clustering and representation learning to generate interpretable visualizations. This function uses self-organizing maps with probabilistic clustering assignments and includes a variational auto encoder extended to time-series probabilistic clustering.
Output: • Time series clustering and forecasting (predictions of target variables) • Generated interpretable visualizations of the clusters • Trained DL model (T-DPSOM) from this function

Table 4.49 Function: Deep spatio-temporal network with data fusion

Function: Deep spatio-temporal network with data fusion
Input: • Multidimensional numerical data set that contains geolocations generated from localization functions with user ids, maximum uplink and downlink throughputs • Model hyperparameters
Description: This function refers to a DL-based method to estimate the network demand in terms of maximum uplink and downlink throughputs. This DL model employs convolutional-based dense networks to model both nearby and distant spatial dependencies between regions of the cities and includes several branches for fusing external data sources of different dimensionality. This DL model uses three different dense networks to model various temporal properties consisting of closeness, period, and trend.
Output: • Estimates of network demand (maximum uplink and downlink throughputs) • Trained DL model from this approach

Table 4.50 Function: Reinforcement Learning/Multi-armed Bandit

Function: Reinforcement Learning/Multi-armed Bandit
Input:

• Multidimensional numerical data set that contains geolocations generated from localization functions with user ids and radio environment maps • Model hyperparameters
Description: This function refers to a ML-based method to provide improved scheduling decisions, to enhance user throughput, and a better bandwidth utilization. This function includes epsilon greedy, decayed epsilon greedy, softmax exploration, and upper confidence bound as solution strategies.
Output: • Scheduling decisions • Trained DL model from this approach

4.2.5.2 Training and Deployment Pipelines

This section presents different training and deployment pipelines based on the ML functions described above. The ML/DL modelling options for the LOCUS platform remain open, as the various deployment pipelines will have to cater to the needs of the UCs described in D2.1 [1], which are currently under development. Indicative pipelines allowing for an offline stream - possibly in a parallel to the LOCUS platform inference/prediction stream and even outside of the LOCUS platform will be considered. Online model training may also be within the LOCUS project scope, if it is deemed necessary for the purposes of the UCs deployed. Currently, some indicative -but subject to change-examples of ML pipelines are presented in the following subsections.

4.2.5.2.1 Training ML/DL models One-off (Offline/batch training) and Inference (Predictions) on the fly

In this approach a pre-trained ML model (i.e. a model both trained and evaluated offline using a pre-determined dataset) is used.

Example 1: Network demand forecasting (SNM-UC1: Knowledge building for network management, described in detail in [1])

Figure 4.2 exemplifies the case of offline training and inference on the fly for this UC.

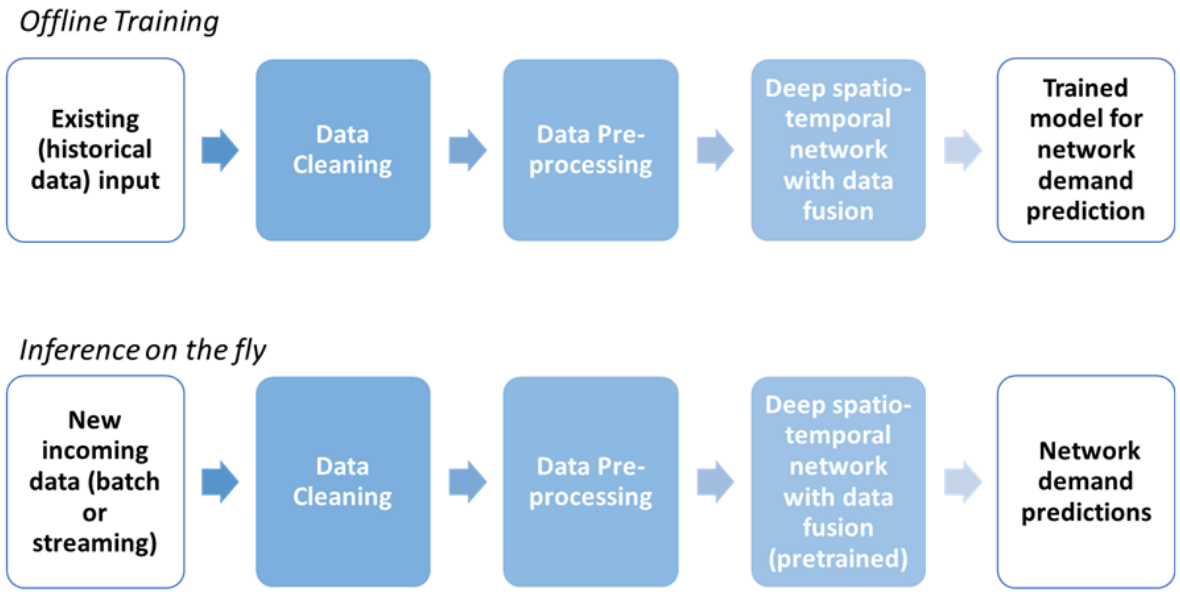


Figure 4.2 Network demand forecasting ML pipeline

Offline Training: Existing datasets after cleaning and pre-processing are used to train a DL model to estimate the network demand in terms of maximum uplink and downlink throughput. The outcome of this pipeline is a trained DL model for network demand prediction.

Inference on the fly: The new data after cleaning and pre-processing is used as test data for predictions (inference) by the pre-trained DL model. The outcomes of this pipeline are the predictions for network demand (max uplink and downlink throughput).

Example 2: Learning group mobility characteristics using wireless fingerprints (NSE-UC2: Crowd mobility analytics using mobile sensing and auxiliary sensors, as described in [1])

Similarly, for example 2, the ML pipelines are presented in Figure 4.3.

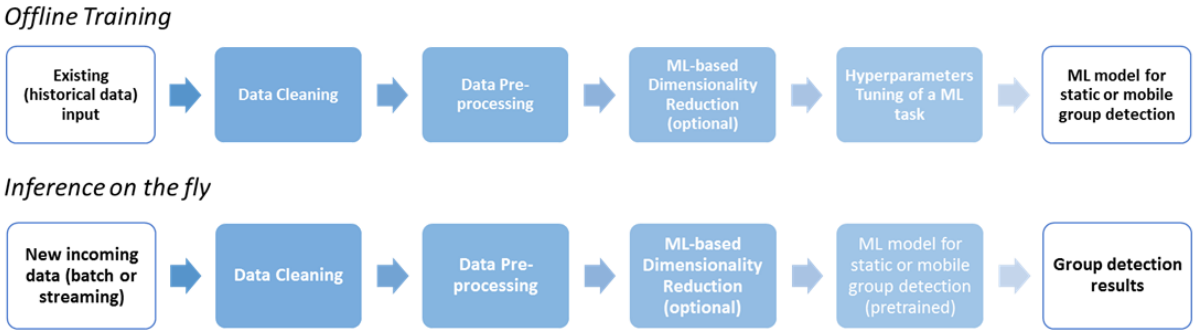


Figure 4.3 ML pipeline for learning group mobility characteristics using wireless fingerprints

Offline Training: Existing datasets after cleaning and pre-processing are used to train ML models to detect groups in the crowd movements. ML-based dimensionality reduction and

hyperparameters tuning of a ML task can be included as optional stages before training. The outcome of this pipeline is a pretrained ML model for the group detection in crowds.

Inference on the fly: The inference pipeline uses a similar set of functions like the training pipeline except that the dimensionality reduction process (if applicable) is predetermined and in the final stage the pre-trained ML model is applied instead of training a new ML model.

4.2.5.2.2 Real-time/Online Training and Inference

In the online learning (also called as incremental learning) the ML models are trained in real-time and tested (inference) on the fly with the new data. A related example follows.

Example: Identifying crowd mobility patterns (NSE-UC1: Flow monitoring and management in large venues and dense urban environment)

The new data after cleaning and pre-processing is split into training and test data. This pipeline can include optional stages like ML-based dimensionality reduction and hyperparameters tuning of a ML task after selecting a specific ML model/algorithm. The training split is used for training a selected ML model and the test split is used for predictions, both happening in real time. The ML functions in this pipeline include a provision for online training/incremental learning from the new data. In this functionality, the outcomes are identified clusters from the crowd movements. The described pipeline is presented in Figure 4.4

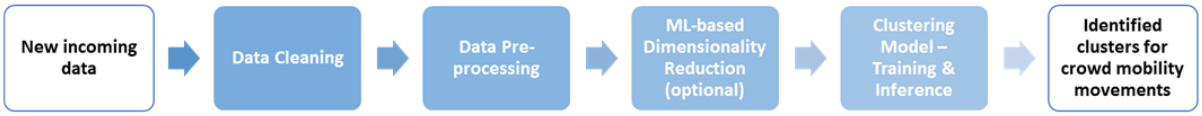


Figure 4.4 ML pipeline for Identifying crowd mobility patterns UC

4.3 Functions and Components Overview

Starting from the definition of the various LOCUS functions provided in the previous sections, Figure 4.5 shows a comprehensive overview of how these functions are positioned within the LOCUS platform. The figure presents different layers, which map to each of the LOCUS function types identified above: data collection, localization enablers, analytics and machine learning. The LOCUS data store, from a logical perspective, can be seen as the main enabler of data and information persistency and exchange among the LOCUS functions residing in the same or different layers. The location security and privacy functions regulate the access to the data stored in the LOCUS data store, and all of the LOCUS functions can access the LOCUS data store according to their input/output data requirements and specific constraints. The LOCUS data store maintains the different outputs generated by the localization enablers (i.e. the geolocation information) and the ML and analytics functions as part of their data processing. Each of these outputs are used as inputs for other LOCUS functions (depending on their data

requirements), or either exposed to external applications as part of their service requests. Indeed, Figure 4.5 shows a top-level application layer which is meant to host the LOCUS applications, which are basically mapped to the Smart Network Management and vertical application use cases described in Deliverable D2.1 [1].

These applications leverage on the set of services that LOCUS exposes through the localization analytics as a service APIs to request for specific localization services, which are in turn mapped to the LOCUS analytics functions outputs. These APIs are under the control of the LOCUS MANO, which takes care to translate the service requests coming from Smart Network Management and vertical applications into LOCUS analytics, ML and localization enabling functions to be instantiated, chained and configured automatically to satisfy the specific application requirements. The LOCUS MANO also manages the related data exposure (as part of the service response) towards the Smart Network Management and vertical applications by leveraging on the LOCUS security and privacy functions.

More details on the overall LOCUS architecture are provided in Section 8, where Figure 4.5 is further elaborated in a system wide approach.

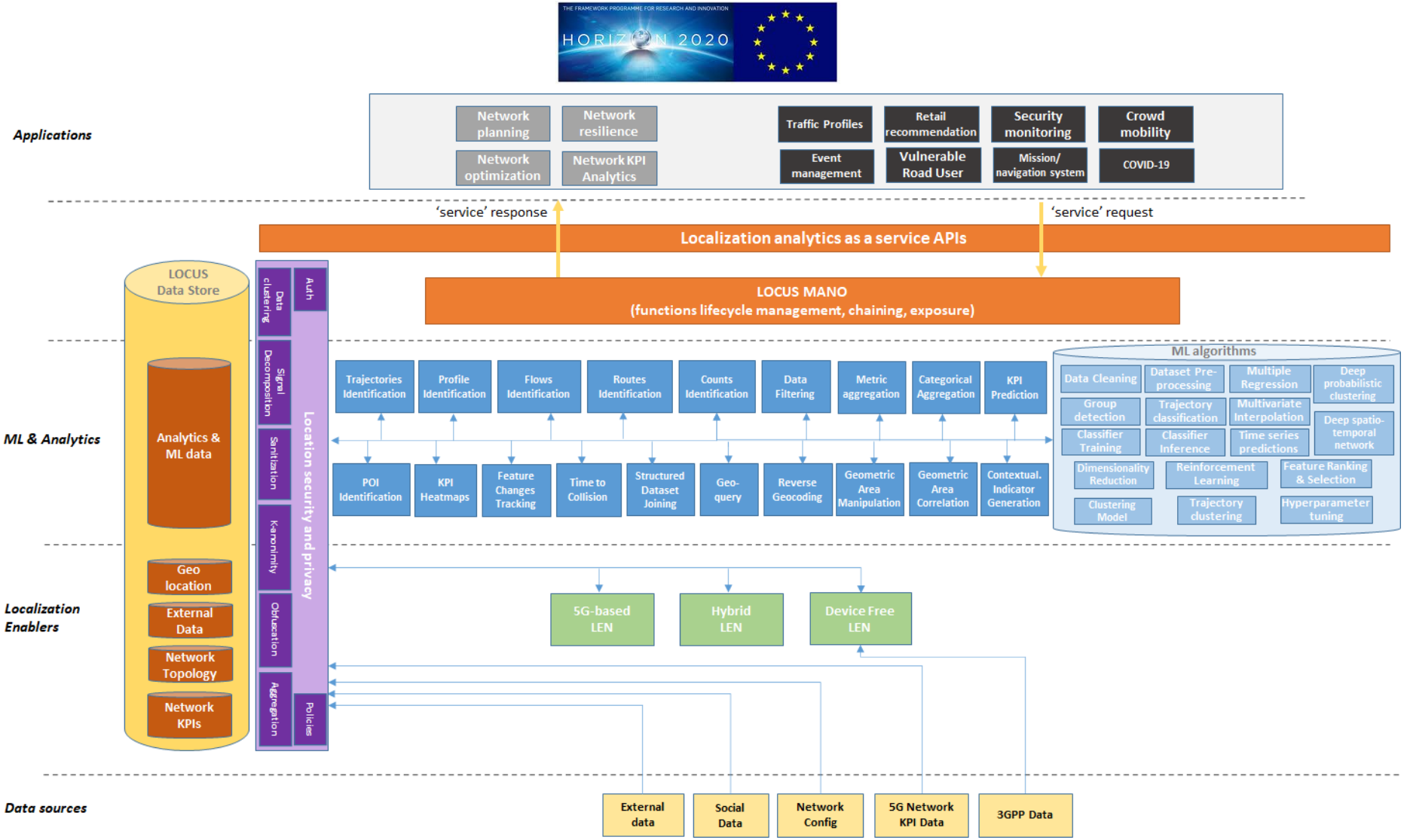


Figure 4.5 LOCUS functions and components overview

4.4 Functional Decomposition

In the previous sections, the localization analytics functions were enumerated along with various data collection and management, security and privacy functions, and indicative ML pipelines that cater to the training and inference process. In fact, the localization analytics exposed as a service to the external to LOCUS platform applications can be decomposed in multiple instantiations of the functions detailed above, properly chained in order to produce the desired result.

This result will be further explored by the Application Layer functions outside of the LOCUS platform, for which there is a short description in Section 5 with respect to the proposed UCs. Additionally, a description on how these analytics are exposed as services can be found in Section 6. In this section, indicative chained functions are presented as examples. However, it will be further detailed and enhanced in the final version of the LOCUS Architecture deliverable (D2.5).

4.4.1 KPI (Network Demand) Prediction Function – Inference on the fly

This function allows for the prediction of a selected KPI, in this case network demand (i.e. maximum uplink & downlink throughput). In Table 4.51 an example decomposition of this function is presented. While the results to be exposed by the LOCUS platform in Table 4.51 are inferred through an already trained model, we may also assume a training function process as a suggested option under consideration.

Table 4.51 Network Demand Prediction decomposition (Inference pipeline)

Step	Scope	Function	Parameters	Notes
1	Data Management[8]	Query Structured Data	New batch historical Location Data (Lat, Lng, EntityID, Time, targeted KPI),	Targeted KPI: maximum uplink and downlink throughputs
2	ML	Data Cleaning	Imputation/interpolation of missing values, removing outliers, correcting inconsistencies in geo-	

[8] Data Management Functions are detailed in Section 8.

			spatial locations and time inputs.	
3	ML	Dataset Pre-processing	Feature Scaling, Preparing input data for fusion: extracting temporal functional regions based on POIs, calculating crowd counts, inferring the day of week to check the cellular usage on that particular day.	
4	ML	Deep spatio-temporal network with data fusion	Model hypermeters set at training phase (such as number of convolutional layers, RELU and dense blocks, learning rate, optimizer, number of training epochs, batch size etc.)	Function uses pretrained model
5	Data Management	Persist Structured Data	Estimates of network demand (maximum uplink and downlink throughputs) in the future.	

Furthermore, in this decomposition it is assumed that new predictions are not time critical and data can be provided in batches through the LOCUS data store. Alternatively, steps could involve a real-time, streaming data acquisition through a data collection function, also persisting data and therefore an alternative Step 1.

4.4.2 POI Identification Function - Online training option

In the following table, the various steps within the LOCUS platform for the detection of POIs is presented as a series of instantiated functions. In more detail, this function can be decomposed into functions that use historical data from the persistence entity, pre-processes them, then a clustering process through the relevant function takes place, where either a hyperparameter tuning takes place (step 4.2), or preset parameters for clustering are set to deliver the clustering result that denotes the POIs. These results are then mapped to a

geometric area and the results are again stored either for use by other components or to be made available to 3rd parties and SNM applications.

Table 4.52 POI Identification function decomposition

Step	Scope	Function	Parameters	Notes
1	Data Management	Query Structured Data	Historical Location Data (Lat, Lng, EntityID, Time)	
2	ML	Data Cleaning	Imputation of missing values	
3	ML	Dataset Pre-processing	Min-Max Scaling, Time-Multiplication	
4.1	ML	Clustering Model – Training and Inference	Density-based clustering (DBSCAN/OPTICS), optimal/set hyperparameters	
4.2	ML	Hyperparameter tuning of an ML task	Grid search through hyperparameter space set (i.e. Epsilon, min # of points), Target KPI (e.g. Silhouette), Target ML function: step 4.1.	
5	Persistence	Transform Structured to Unstructured	Clustering result as input, Geometric Area Representation	
6	Persistence	Persist Unstructured Data	Geometric Areas Collections, Index on Boundary, Center, Time	

5 LOCUS Application Layer Functions

This chapter presents the main capabilities and features of the LOCUS application layer, in terms of functions provided and high-level localization analytics requirement to be fulfilled by the LOCUS platform.

The LOCUS applications are linked to the Smart Network Management and Vertical use cases defined in D2.1 [1], and are the main consumers of the localization analytics produced within the LOCUS platform, that are indeed requested and consumed through the Localization Analytics as a Service APIs.

The LOCUS applications are therefore classified into two main classes (Smart Network Management and Vertical/3rd party), and three major categories:

- People Mobility & Flow Monitoring
- Network-Assisted Self-Driving Objects
- Smart Network Management based on Localization

The following subsections provide details about each of the application that will be developed in LOCUS as a way to exploit and validate the localization analytics exposed by the platform. These descriptions follow a common template in the form of table, as shown in Table 5.1, and aim at identifying the main required analytics from LOCUS and the application logic. It is worth to mention that the LOCUS platform aims at providing a new approach toward generalized and unified localization analytics as a service, suitable and capable to accommodate other type of applications.

Table 5.1 LOCUS application description template

<app name>	
Category	<Smart Network Management based on Localization/Network-Assisted Self-Driving Objects/People Mobility & Flow Monitoring>
Description	<textual description of the application>
Required Analytics from LOCUS	<what is required from LOCUS>
Application Logic and functions	<what the application does and generates as output>

5.1 Smart Network Management

The Smart Network Management applications refer to the related use cases defined in D2.1 [1], as follows:

- Network KPI Analytics as part of the Knowledge Building use case (SNM-UC1). While the knowledge building is performed within the LOCUS functional block "Localization & Analytics for Smart Network Management (SNM)", the relevant App exposes the outputs of this functional block for network monitoring purposes.
- Location aware network planning in 5G (SNM-UC2)
- Location aware network optimization in 5G (SNM-UC3)
- Location aware network Location aware network resilience in 5G (SNM-UC4)

5.1.1 Knowledge Building for Network Management (SNM-UC1)

Table 5.2 Network KPI Analytics App

Network KPI Analytics App	
Category	Knowledge Building
Description	Monitor network KPI/measurement data in conjunction with location information for the extraction of meaningful and highly informative insights.
Required Analytics from LOCUS	• Collection of Geo-Area (POI identification) results from LOCUS for defined in App area (e.g. mall) at regular intervals • Collection of Geo-Path (Path identification) results from LOCUS for defined in App area (e.g. mall) at regular intervals • Collection of selected KPI values from LOCUS for specified time-frame and area • Collection of selected KPI values from LOCUS for specified time-frame, geospatially correlated to Paths and POIs detected in area
Application Logic	• Creation of images (i.e. heatmaps and other analytics) for network KPI analytics

5.1.2 Location Aware Network Planning in 5G (SNM-UC2)

Table 5.3 Network planning App

Network planning App	
Category	Network Planning
Description	Collect relevant information and provide planning recommendation including the choice of sites and configurations under constraints such as cost and EMF.

Required Analytics from LOCUS	• Collection of heatmaps and other analytics provided by SNM-UC1 • Collection of additional KPIs from LOCUS for specified time frame and area
Application Logic	• Collection of planning scenario information: up-to date information on network topology, including legacy deployment, candidate sites for installing 5G Node Base stations (BS), target goals (e.g. performance, EMF minimization, Capital expenditures, etc.) • Creation of a list of chosen sites to host 5G-Node BS, the configuration of each site in terms of antenna elements/settings, and the costs/performance/EMF/localization values

5.1.3 Location Aware Network Optimization in 5G (SNM-UC3)

Table 5.4 Network optimization App

Network optimization App	
Category	Network Optimization
Description	Collect relevant information and provide network parameter setting changes that optimize predefined goals such as capacity or end-user Quality of Service (QoS).
Required Analytics from LOCUS	• Collection of Performance Management (PM) and Configuration Management (CM) network data • Collection of up-to date information on network topology • Collection of heatmaps and other analytics provided by SNM-UC1
Application Logic	• Collection of network optimization goals • Creation of network optimization recommendations • Creation of network optimization policies and configuration changes towards the OSS

5.1.4 Location-Aware Network Resilience in 5G (SNM-UC4)

Table 5.5 Network optimization App

Network resilience App	
Category	Network Resilience

Description	Collect relevant information and provide fault detection, identification of root causes and corrective actions on the network.
Required Analytics from LOCUS	• Collection of Fault management (FM), Performance Management (PM), and Configuration Management (CM) network data • Collection of up-to date information on network topology • Collection of heatmaps and other analytics provided by SNM-UC1
Application Logic	• Creation of network dysfunction alarms towards the OSS and/or a network monitoring interface • Creation of change recommendations based on root cause analysis • Creation of network configuration changes or any corrective action to be performed on the network

5.2 Vertical Apps

Similarly, the vertical Apps revolve around new services utilizing People Mobility & Flow Monitoring and Network-Assisted Self-Driving Objects analytics.

5.2.1 Flow Monitoring and Management in Large Venues and Dense Urban Environments (NSE-UC1)

Table 5.6 Security Monitoring App

Security Monitoring App	
Category	People Mobility & Flow Monitoring
Description	In a security monitored indoor area, i.e. a mall, an airport or another venue that also includes authorized and unauthorized areas, people mobility is tracked at all times. The venue security will receive alerts, including the existence of unauthorized persons and unauthorized/dangerous/suspicious congregations of people in points of interest.
Required Analytics from LOCUS	• Collection of Geo-Path (Path identification) results from LOCUS for defined in App area (any indoor area) at regular intervals • Collection of User location from device sensors and/or LOCUS (given user permission) at regular intervals

	• Detection of users in unauthorized area
Application Logic	• Collection of external metadata related to venue/area maps with the ability to define authorized and unauthorized areas, defined by user through the App • Real-time map visualization of the authorized and unauthorized areas. • Creation of alert specifications/conditions received by venue security.

Table 5.7 Flow Management - Venue Admin App

Flow Management - Venue Admin App	
Category	People Mobility & Flow Monitoring
Description	In an indoor area, i.e. a mall, an airport or another large venue, people mobility is tracked at all times. The venue administrator and/or network owner will instruct the LOCUS platform to monitor and predict immediate flow patterns so as to track crowd flows. This way the venue administrator can make decisions for e.g. sending more personnel to assist in crowd management areas, or manage the personnel resources in the various shops so as to ensure quick customer service and decongestion. This app could raise alerts based on the predicted paths of the area visitors based on criteria set by the users.
Required Analytics from LOCUS	• Collection of Geo-Path (Path identification) results from LOCUS for App defined area (any indoor area) at regular intervals and predicted movements in the near future. • Collection of people counts in areas and predictions in the near future.
Application Logic	• Collection of external metadata related to venue, area maps with the ability to define paths, maximum people density per area, etc. defined by user through the App. • Real-time map visualization of the people flow and the projected movements. • Creation of alert specifications/conditions received by venue administrator.

Table 5.8 Flow Management for Verticals App

Flow Management for Verticals App	
Category	People Mobility & Flow Monitoring
Description	Assuming an indoor/outdoor area, i.e. a mall, an airport or other large venue, LOCUS platform people mobility and flow monitoring information are offered to other parties so as to enable vertical sectors; e.g. in the retail sector, assuming multiple buildings/shops (e.g. boutique stores, diners/cafés, recreational spaces), a recommendation App utilizes the location-based analytics, i.e. visitor paths and points of interest based on mobility patterns, and gives recommendations to users with the installed retail App as they move within a specified area.
Required Analytics from LOCUS	• Collection of Geo-Path (Path identification) results from LOCUS for App defined area (any indoor area) at regular intervals • Collection of user locations from device sensors and/or LOCUS (given user permission) at regular intervals (This can also be done through the user device at the App layer).
Application Logic	• Collection of external metadata related to venue, area maps with the ability to define shops/areas/paths, metadata defined by the user through the App. • Collection of information related to user profile and user history for refining the vertical service in question through the App (assuming user permissions). • Creation of preferences/conditions set by the user. • Vertical-related functionality; for our example, ML-based notifications for next best retail recommendation based on localization analytics and map visualisation of recommended shop.

5.2.2 Crowd Mobility Analytics Using Mobile Sensing and Auxiliary Sensors (NSE-UC2)

Table 5.9 Crowd mobility App

Crowd mobility App	
Category	People Mobility & Flow Monitoring
Description	Provides a presentation layer for the analytics taking place inside LOCUS, where crowd mobility is monitored (i.e. crowd

	size, existence of groups) and events detected. The App provides a wireless scanning system to detect static or mobile people groups in indoor or outdoor environments. It can detect not only static groups but also moving groups with a multi-phased approach based on noisy wireless Received Signal Strength Indicator observed by multiple wireless scanners.
Required Analytics from LOCUS	• Collection of Geo-Path (Path identification) results from LOCUS for defined Area at regular intervals • Collection of User location from sensors and/or LOCUS at regular intervals • Group detection in short time intervals (minutes) • Long-term (months) group linkages detection
Application Logic	• Monitoring/presentation layer for acquired analytics

5.2.3 Vulnerable Road User (NSE-UC3)

Table 5.10 Vulnerable Road User App

Vulnerable Road User App	
Category	People Mobility & Flow Monitoring, Network-Assisted Self-Driving Objects
Description	Alerts Host Vehicle (HV) that a Vulnerable Road User (VRU) is approaching
Required Analytics from LOCUS	• Collection of User location from device sensors and/or LOCUS (given user permission) at regular intervals • Collection of Geo-Path (Path or Route identification) results from LOCUS for defined in App area at regular intervals • Collection of Time To Collision information
Application Logic	• Collection of User Profile metadata (e.g. HV, VRU) from App registration • Collection of external metadata related to roads, public transport routes etc. • Creation of Alert specifications/conditions by App user

5.2.4 Logistics in a Seaport Terminal Using Automated Guided Vehicles (NSE-UC4)

Table 5.11 Mission/navigation system

Mission/navigation system	
Category	Network-Assisted Self-Driving Objects
Description	Uses position and velocity information provided by the 5G network to determine the next moves of Automated Guided Vehicles (AGV) on the planned path, i.e. handles the planning of the shuttling tasks, determines the trajectory of the AGV, computes and transfers the next motor command to the AGV.
Required Analytics from LOCUS	• Collects positioning information at regular intervals for each AGV from the LOCUS platform
Application Logic	• AGV motion controller function: this function controls the navigation and movements of the vehicle in the area. It receives information from the trajectory planning function and from the positioning collection function and, based on this information, it computes the next movement step • AGV task planning function: this function coordinates the activities in the seaport warehouse area and takes care of assigning the vehicles to the different missions depending on their status and their current position respect to the freight to shuttle. This function handles also the relational database containing the freights inventory and the vehicle data. It will make use of a rule-based expert system and a relational database based on mySQL • AGV trajectory planning function: this function is used to determine the path the vehicle must follow in the test area first to reach the freight to pick and then the final destination. The algorithm will receive a map (see next function) reporting the areas were the vehicle can and can't go through. These areas include also the other freights placed in the seaport area whose position is pre-determined • Map manager: the map is built on the fly based on the information about static object in the area, and position and size of freights contained in the inventory in the relational DB

5.2.5 Transportation optimization based on identification of traffic profiles (NSE-UC5)

Table 5.12 Traffic Profiles App

Traffic Profiles App	
Category	People Mobility & Flow Monitoring
Description	Offers a presentation layer for Path profile analytics based on velocity characteristics and an alert functionality.
Required Analytics from LOCUS	• Collection of Traffic Profile results from LOCUS for defined in App area at regular intervals
Application Logic	• Collection of external metadata related to roads, avenues, public transport routes etc. • Creation of Alert specifications/conditions by App user • Tracking LOCUS information and offer alerts as defined by App user • Real-time map visualization of alerts in area

5.2.6 Positioning and Flow Monitoring for Controlling COVID-19 (NSE-UC6)

Table 5.13 COVID-19 Contact Tracing and Proximity App

COVID-19 Contact Tracing and Proximity App	
Category	People Mobility & Flow Monitoring
Description	Given an identified case of COVID-19 infection, it traces back the persons to have potentially been in proximity with the positive case within a certain number (to be set) of previous hours/days.
Required Analytics from LOCUS	• Record contact events, and provide timely information status on past contacts in related to their COVID-19 situation • Monitor and classify mobility behaviour in accordance with current quarantine and health protocols
Application Logic	• Trigger a proximity event whenever proximity/contact criteria are detected by using the available geolocation information • Provide timely updates on current quarantine and health protocols, and trigger a violation event whenever a quarantine is violated or a mobility behaviour is classified as risky, illegal, etc.

Table 5.14 COVID-19 Monitoring Epidemiological Risk Flow App

COVID-19 Monitoring Epidemiological Risk Flow App	
Category	People Mobility & Flow Monitoring
Description	Estimates Risk factors and their spatiotemporal evolution using epidemiological data combined with the flows of people moving from one area to another area.
Required Analytics from LOCUS	• Monitor people flow from area to area (area dimension depending on the available data, ranging from tiles to census areas) on time frame basis (ranging from every hour to daily basis depending on the available data)
Application Logic	• Combine aggregated mobility data from the census areas with epidemiological data in those area • Provide a spatiotemporal evolution of epidemiological risk based on data aggregation • Predict the risk of contagion in different areas

6 Services

As already anticipated in the previous chapters, LOCUS is conceived to offer localization analytics as a service to the Smart Network Management and 3rd party vertical applications. In practice, this translates into the capability to expose services towards the Application Layer that allow the LOCUS applications to access and consume localization related data, as well as analytics functions and ML model predictions according to the specific Smart Network Management or 3rd vertical requirements.

Therefore, two main categories of services are defined, and as a consequence exposed by the LOCUS platform:

- Access to localization related data, through HTTP REST services that allow to both consume existing data, as well as possibly push new data into the LOCUS platform (according to the security and privacy requirements and constraints regulated by the LOCUS APIs)
- Access to analytics functions and ML model predictions, with two main options: i) HTTP REST services to access analytics and ML functions and consume their outputs based on regular request/response mechanism, ii) message bus based to enable LOCUS applications to consume streams of data generated within the LOCUS platform by localization analytics services.

The following subsections provide a table-based description of these localization related services exposed by LOCUS, including high level communication diagrams to show how LOCUS applications can generally interact with the specific service offered. For the message bus based service exposure, LOCUS assumes the use of Kafka [23], as it can be considered as the de-facto standard solution for publish/subscribe and processing of streams of data in a production-ready and scalable way.

6.1 Expose CRUD operations on collection of structured or unstructured data

Table 6.1 Service: Expose CRUD operations on collection of data

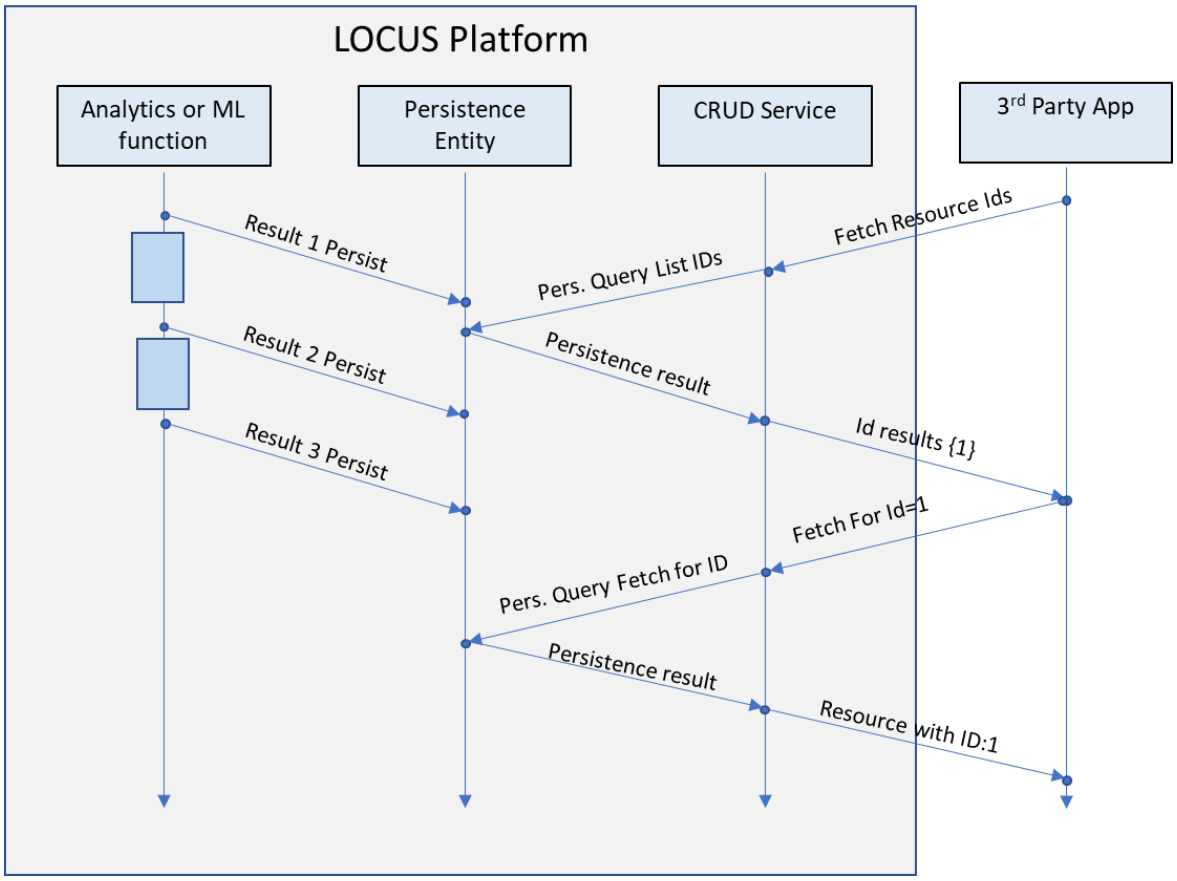
Service: Expose CRUD operations on collection of data
Input: • Target Collection that will be exposed as a CRUD REST API • Access Rights for interfacing with this collection (e.g. only read)
Description: A CRUD service (Create, Read, Update, Delete) is a generic function collection that is used in conjunction with HTTP methods and REST specifications for stateless access of data entities and resources.

In many cases of LOCUS function instances, the function interacts with the persistence entity by storing its produced result. This, along with the various indexing and lookup capabilities, can then dictate the query capabilities that may be used by a REST-based CRUD service and can be accessed by a 3rd party app in order to interact with these generated results. The LOCUS platform is also responsible for the enforcement of appropriate user access control rules for the specific instance of the service function.

Result:

The result of this service will be a microservice entry that will be deployed in the edge of the LOCUS platform and it will provide the CRUD capabilities to 3rd party applications.

Communications Diagram:

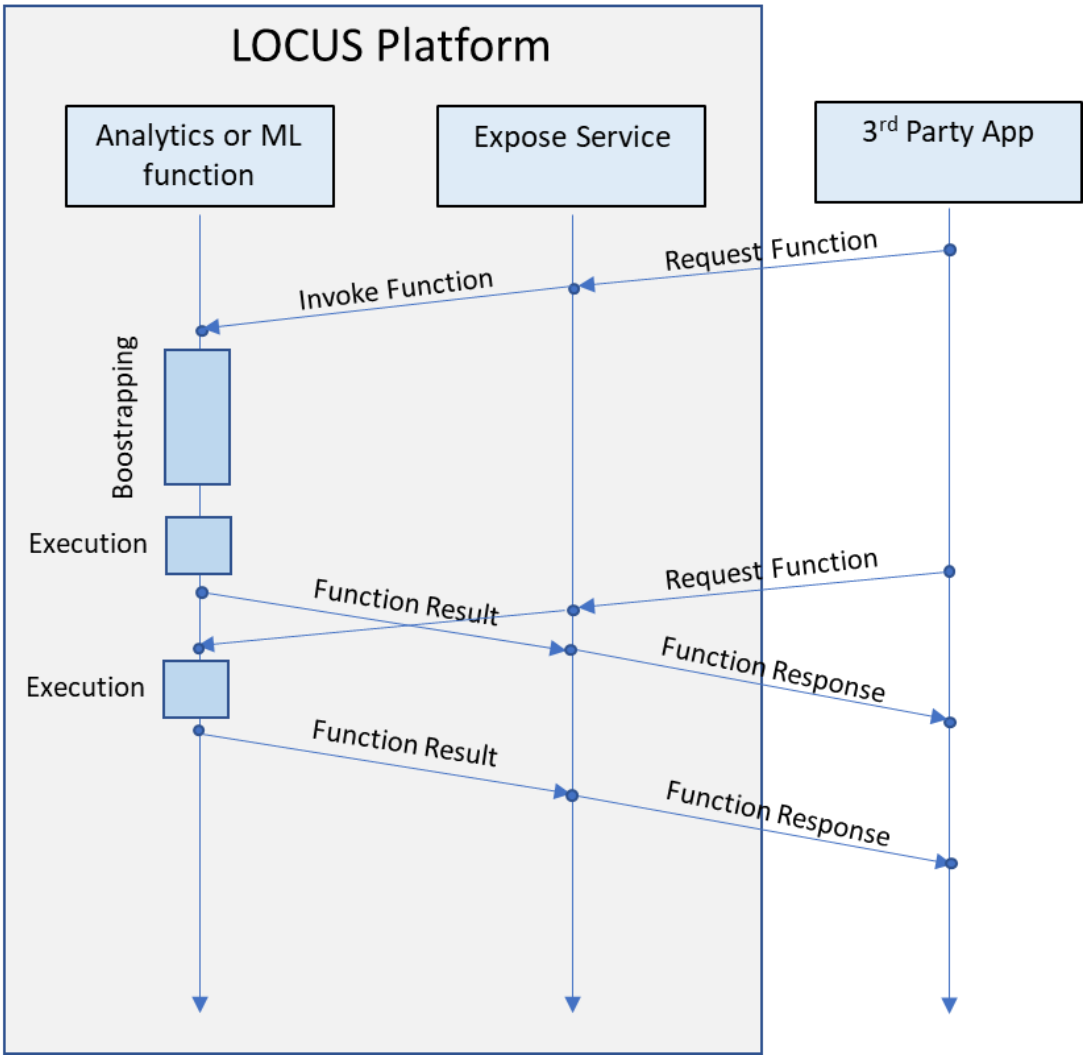


6.2 Expose Analytics Function, ML model predict or High-Level Function as an on-demand REST service

Table 6.2 Service: Expose Analytics/ML function as an on-demand REST service

Service: Expose Analytics/ML function as an on-demand REST service
Input: • Identifier to the internal LOCUS Analytics or ML model predict function or High-Level Function • Access rate restriction/configuration • (Optional) Transformation function for the output of the function
Description: Direct interfacing with the result of an analytics or ML function is also one of the predefined capabilities of the LOCUS platform. This bypasses the need for a persistence entity for mission critical and time sensitive results. In order to define a function exposition endpoint, we need to identify the intra-LOCUS analytics or ML function instance that we wish to expose. This function instance will be linked to an input and output data types. The aforementioned will get inherited by the REST API as the definition for the required request and response body. Considerations must also be made for the access of this endpoint as the invocation of an analytics or ML function can be very demanding in terms of resource requirements. Another (optional) aspect of this service is the definition of a transformation of the data type to a reduced, easier to consume form. This is defined by the output transformation function.
Result: A REST API that will take as input (e.g. in POST body) the requirements for the invocation of the analytics function (or the input data that are required by the model predict function) and it will produce a result as the output type of the analytics function or a structured dataset.

Communications Diagram:



6.3 Expose Analytics Function, ML model predict or High-Level function as a Kafka Topic

Table 6.3 Service: Expose Analytics/ML function as a Kafka Topic

Service: Expose Analytics/ML function as a Kafka Topic
Input: • Identifier to the internal LOCUS Analytics/ML model predict or high-level function • Definition of the Kafka topic and Kafka-related configuration parameters • (Optional) Transformation function for the output of the system function
Description: In a publish-subscribe based approach, another method of exposing the output of one of LOCUS functions is the production of a Kafka topic. Apache Kafka [23] is a publish-subscribe

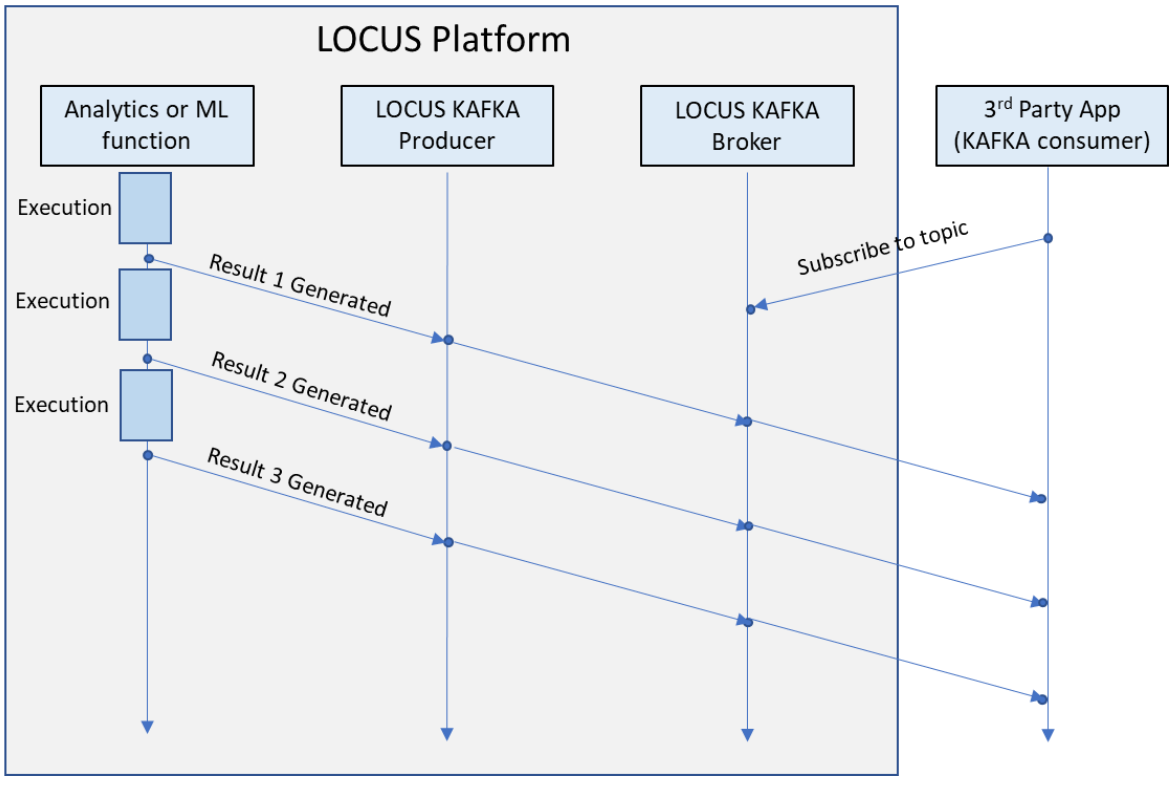
messaging system that sends messages between processes, applications, and servers assigned to a topic (category). Applications may connect to this system, where the broker handles all requests from clients, the producer sends records to a broker and the consumer consumes batches of records.

It is defined in the LOCUS architecture that we will include a Kafka bus for internal and external interfacing with streaming data. In order to expose the output of a system function in a Kafka topic we need to define the topic and some specifications related to the Kafka queue, identify the function from inside the function instances of the LOCUS platform and link it with a Kafka Producer. An additional (optional) feature is a transformation function at the Kafka Producer's layer in order to provide the consumer with the appropriate data structure for the service.

Result:

An Internal Kafka producer and a Kafka topic on the LOCUS message bus, which will provide a message for each execution of the denoted system function.

Communications Diagram:



7 Mapping to existing standards

This section presents a small introduction of the existing standards and attempts to contextualize the work being done within the LOCUS project with respect to them. In this aspect, Section 7 is connected with the System Architecture that is described in Section 8.

7.1 3GPP System Architecture for 5G

The 3GPP System Architecture specification group (SA2) has done extensive work in defining the architectural model for localization services in 5G-NR. The enhanced LoCalization Services (eLCS) specification work was carried out in Release 16 and the phase 2 of eLCS (eLCS-ph2), targeting high accuracy positioning for Industry IoT, will be conducted in Release 17. The architectural model builds on from the 4G LTE releases and supports both emergency/regulatory services and commercial applications.

The main specification document for eLCS architecture is TS 23.273, titled “5G System (5GS) Location Services (LCS)” [28]. The 5G location services related to positioning use cases are detailed in TR 22.872 [29] and asset tracking use cases are detailed in TR 22.836 [30]. The positioning accuracy requirements for vertical industries are captured in the Annex B of TS 22.261 [31]. Broadly speaking, 3GPP SA2 is defining the architecture framework for localization use cases and requirements captured in SA1.

There are three basic types of location requests supported in the 3GPP eLCS architecture. The first is termed as Network Induced Location Request (NI-LR), which is initiated by the serving AMF for a UE for an emergency or regulatory requirement. This type of request can override any privacy settings related to the particular UE. The second type is named as Mobile Terminated Location Request (MT-LR). This is applicable to commercial services and can originate from an external client or an internal network function or an application function of the Public Land Mobile Network (PLMN). The request is made to the PLMN for the location of the target UE. The privacy settings for this client/NF request by the UE is always checked and the request is rejected if it is against the privacy settings. The third type is named as Mobile Originated Location Request (MO-LR). This is used when a UE sends a location request to the serving PLMN for location related commercial application concerning that UE.

The location requests can also be categorised as immediate requests and Location Deferred Requests (LDR). For the immediate location requests, the location response is expected within a short time-frame that is specified in the QoS for the service. All of the above three request types (NI-LR, MT-LR and MO-LR) can be formulated as an immediate request. The LDR however, can be formulated only for the MT-LR type. The external client or NF that makes the LDR expects a location response only when a certain event is triggered (like entering or leaving a certain area) or with a time periodicity.

The 5G system positioning architecture for the non-roaming scenario is depicted in Figure 7.1 below, taken from [28].

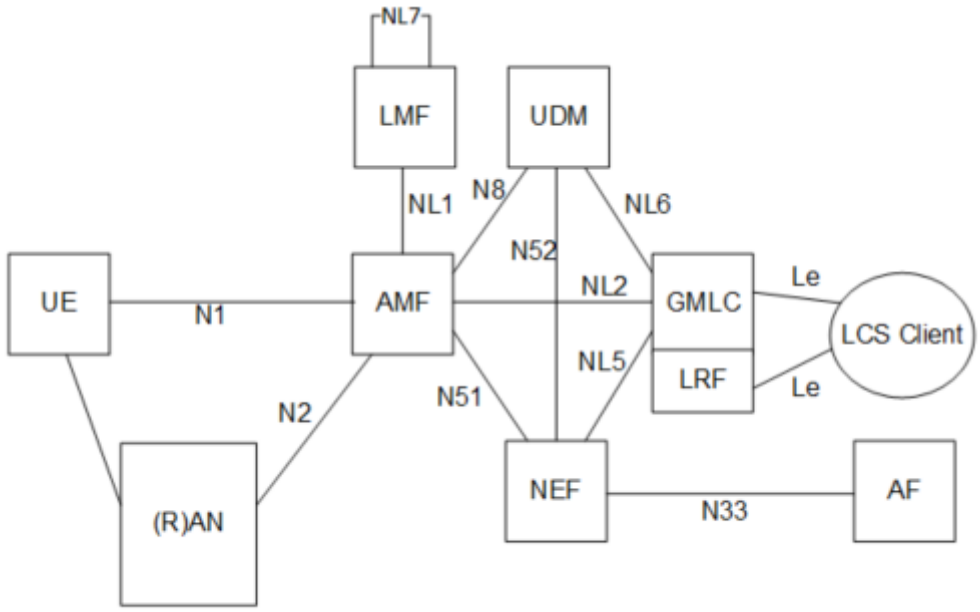


Figure 7.1 Non-roaming reference architecture for Location Services in reference point representation (from [28])

A typical MT-LR example for the call flow messages within this architecture is as follows:

a. The application function or the network function makes a localization request to the GMLC (Gateway Mobile Location Centre) through the NEF (Network Exposure Function). Equally an external location client can directly make a location request to the GMLC, through the Le interface.
b. The GMLC checks with the UDM (Unified Data Management), if the privacy setting for UE allows this request. If not, the request message flow is terminated. GMLC further checks what is the serving AMF (Access and Mobility Function) for the UE.
c. The GMLC makes the 'Provide Positioning Info Request' to the AMF.
d. The AMF provides the response to the above request to GMLC. GMLC passes down the response, to the level of AF or external client.
e. The AMF issues a 'Network Triggered Service Request' to the UE, through the Non-access stratum (NAS).
f. The UE responds to the request through the NAS.
g. The AMF does the LMF (Location Management Function) selection. This selection is based on the physical closeness to the serving Access Node to the UE, amongst other factors.
h. The LMF instructs for the UE positioning using the Next Generation Radio Access Network (NG-RAN). This can also be the 4G-RAN (LTE), trusted or untrusted non-3GPP radio technology.

i. LMF provides the 'Location_DetermineLocation' response to the AMF.
j. AMF provides the 'Location_ProvidePositioningInfo' response to the GMLC.
k. The GMLC provides the 'LCS service response' to the external client (or the network function through the NEF).

The reference architecture with the Service Based Interfaces (SBI) is shown below in Figure 7.2. With the SBI, every network entity that has an interface line beginning with a dot can communicate with every other similar network entity. With the native virtualisation provided in 5G, these functions can reside anywhere, even in the Cloud.

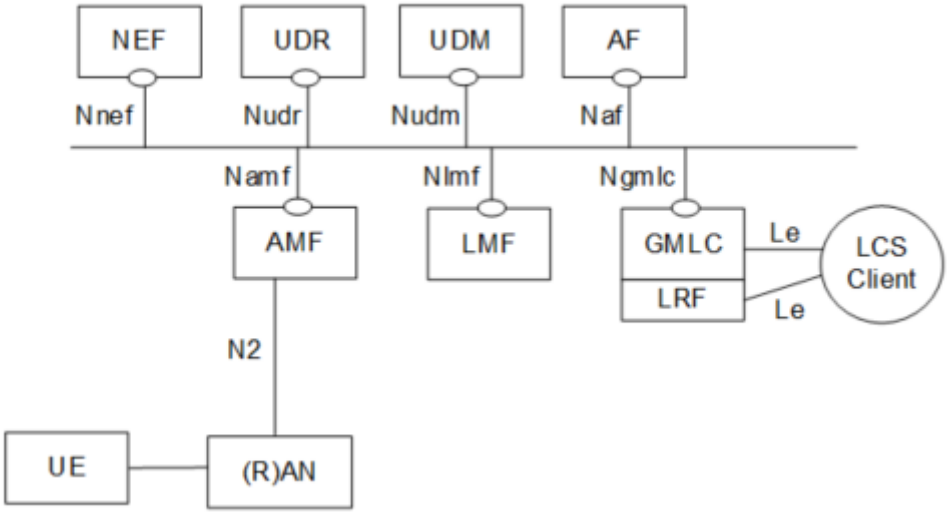


Figure 7.2 Non-roaming reference architecture for Location Services in SBI representation (from [28])

7.2 Relation to ETSI NFV MANO architecture

The ETSI NFV Industry Specification Group (ISG) has been the driver of the network transformation activities in ETSI, being at the core of the NFV technology definition and standardization, starting from the NFV term itself. As shown in the architecture diagram in Figure 7.3, the NFV scope is focused on the lifecycle of the Virtualized Network Functions (VNFs) and the services built by composing them, and with other components generally referred as PNFs (Physical Network Functions), irrespectively of their nature, scope, technology. With NFV, network functions are implemented in software and can run on homogeneous, industry-standard commodity infrastructures. This software can then be moved to, or introduced in different locations in the network as required. NFV simplifies the roll-out of network services, reduces deployment and operational costs and facilitates network management automation.

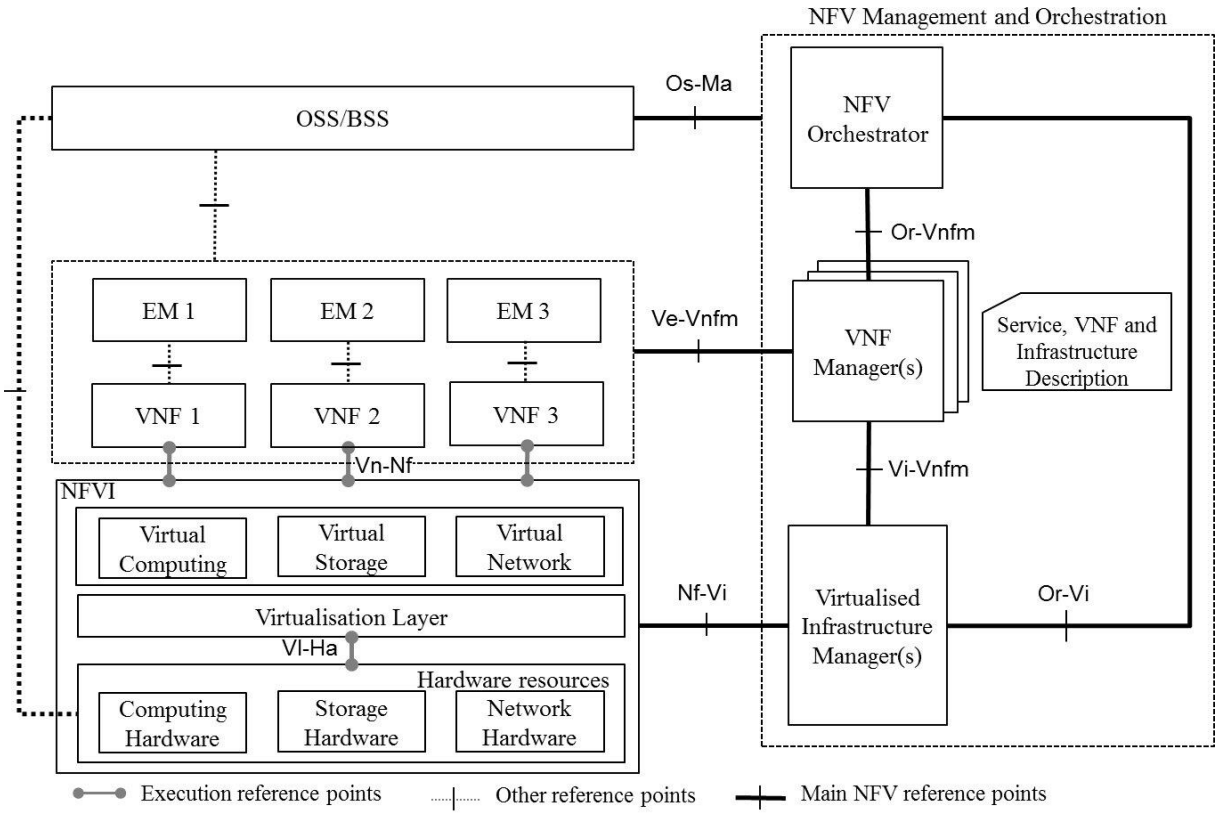


Figure 7.3 ETSI NFV architectural framework

Up to now, the ISG has defined the initial framework and continued in the following phases working on interfaces, information models and testing procedures. Detailed specifications for NFV related workflows, data structures and APIs are now under consolidation in the close-to-finished Release 3. At the current stage, the NFV community has produced a wide set of mature and stable specifications, has consolidated their applicability in the industry, and continues to work towards multi-vendor interoperability and addressing new architectural and application challenges. The available NFV standards of Release 3 are already used in the industry to implement NFV products. NFV was originally conceived to help network service providers in the challenge of cost reduction and agility improvement, and it has resulted to be a key framework to enhance how services are requested and consumed by users. It is a necessary component for next-generation networks, and in particular for 5G.

A central role in the ETSI NFV architecture is taken by the MANO, which covers the lifecycle management of two entities: network services and VNFs. As said, VNFs are network functions implemented in software and ready to be deployed and operated over virtualized infrastructures (e.g. OpenStack [20] based infrastructures), while the network service is the interconnection of VNFs which chained together build a standalone service application logic.

It is composed by three main building blocks: the NFV Orchestrator, the VNF Manager (VNFM) and the Virtualized Infrastructure Manager (VIM). The VIM takes care to manage the

virtualized infrastructure, that could span across several locations (e.g. core data center locations and edge locations), and offer to the other MANO components specific APIs to create, update and delete virtualized resources for running VNFs (e.g. virtual machines, containers, virtual networking objects, etc.). It is normally implemented through Infrastructure as a Service (IaaS) platforms such as Openstack. The VNFM is responsible for the lifecycle management of each VNF building a network service, and coordinates all the phases for instantiation, scale and termination, taking care of their performance and fault management as well. On top of the other components, the NFVO coordinates the lifecycle of the network services, and is responsible to create them as the composition of multiple VNFs by interacting with the VNFM and the VIM. As shown in Figure 7.4, the ETSI NFV MANO enables to make automated the full lifecycle of network services and VNFs, starting from well-defined (and standardized within ETSI NFV) service and function description templates (i.e. the descriptors) and going through standard procedures and APIs for service instantiation, scale, performance monitoring, fault management. In particular, the VNF Descriptor (VNFD) [32] and the Network Service Descriptor (NSD) [33] can be considered as the reference templates that regulate functions and services lifecycle management.

Therefore, the NFV software-enabled function and service lifecycle management provided by ETSI NFV MANO constitutes a new dimension to be considered in the network and service management field, and it has to be:

- Integrated with other end-to-end management tools and system, such those for smart network management within advanced OSS, as well as network slicing in 5G scenarios
- Considered as an essential enabling technology for network automation and simplification, with services to be easily deployed in multiple instances in distributed locations
- Considered as a key target for new operational architectures and deployments where automation of virtualized environment and the services running on top is required.

All of these makes the ETSI NFV framework, and in particular the MANO architecture very relevant for LOCUS, as it offers the APIs, the data models and the workflow logics for building complex services as the combination of individual virtualized functions running in software and deployed over a virtualized infrastructure, either in the form of virtual machines or containers. In particular, the localization services envisaged by LOCUS can be implemented as a specific case of NFV network services, being them the combination of data collection and storage functions, localization functions, analytics functions, machine learning functions to be interconnected according to their data requirements at both input and output sides. On top of these localization services, LOCUS builds specific applications to cover two main areas: Smart Network Management, in support of advanced OSS procedures and logics, and third-

party applications, in support of vertical specific services for self-driving objects, people mobility and flow monitoring.

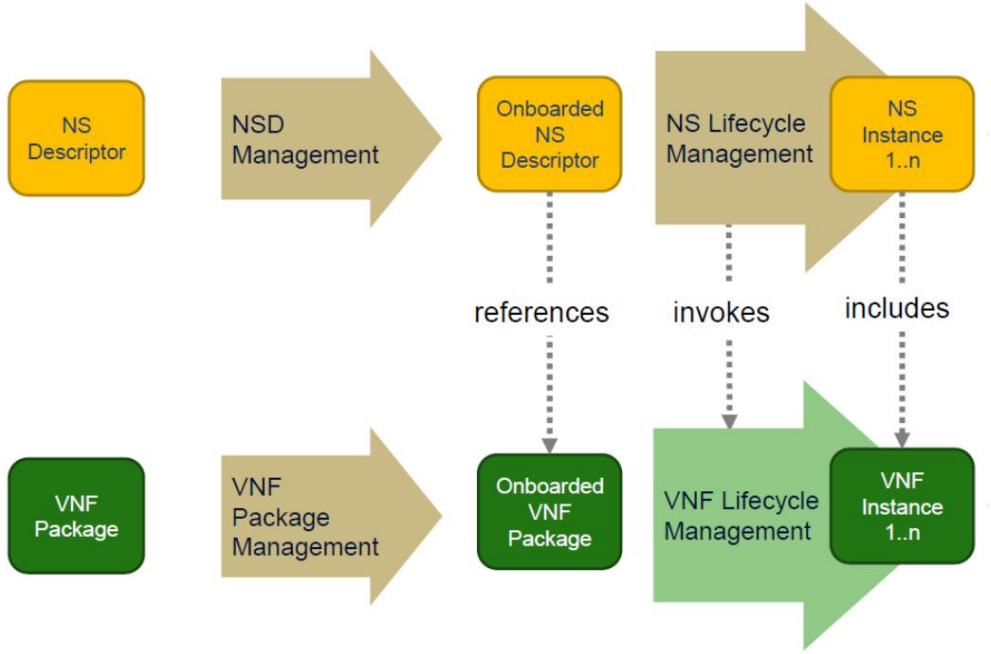


Figure 7.4 ETSI NFV MANO lifecycle management principles (source: ETSI)

Therefore, according to the specific Smart Network Management or vertical application needs in terms of required data, localization scope, localization features, dedicated combinations of LOCUS localization related functions (including analytics and machine learning) should be prepared to compose proper localization services. This means that the network service descriptor (NSD) should be designed, prepared and onboarded in the MANO system according to the requirements of the LOCUS applications

The ETSI NFV MANO can be then considered as a reference approach for managing and coordinating the lifecycle of the LOCUS localization services. Its principles can be reused to achieve a full automation in the deployment and operation of localization services over virtualized infrastructures.

7.3 Relation to ETSI ZSM

The ETSI Zero touch network and Service Management (ZSM) ISG is aiming at defining a new, horizontal and vertical end-to-end operable framework enabling agile, efficient and qualitative management and automation of emerging and future networks and services. ZSM targets a network and service management architecture where all operational processes and tasks (e.g. delivery, deployment, configuration, assurance, and optimization) are executed automatically, ideally with full automation in multi-vendor environments. The existing work from ETSI NFV solves dedicated aspects of network and service management, and has defined

management capabilities for their respective network function and service virtualization domain. On top of this, ETSI ZSM aims to provide a holistic service management concept which, among others, enables the integration of NFV and edge computing management demands with other relevant service management aspects such as data collection, analytics, intelligence. However, NFV-like architectures are still based on monolithic control and orchestration solutions, mostly dedicated to the management of pipelined network services, where the lack of agility in the service lifecycle and operation is still a clear limitation, especially when it comes to fulfilling heterogeneous service constraints posed by 5G services and beyond.

ZSM goes in the direction of overcoming such limitations, and its reference architecture is built around a set of building blocks that collectively enable construction of more complex management services and management functions. The clear identification and separation of management domains provide means to isolate management duties (possibly referring to very heterogeneous technologies), considering boundaries of different nature (technological, administrative, geographical, etc.). Each management domain provides a set of ZSM management services, realized by functions that expose and/or consume a set of end-points. An end-to-end service management domain is a special management domain responsible for the cross-domain management and coordination, which glue all of the single domain management services, functions and end-points.

At the core of the ZSM architecture there are the domain integration fabrics and the cross-domain integration fabric, which facilitate the provision of services and the access to them through the related end-points across the various domains. It also includes specific services for the communication between management functions, which enable the exchange of management data to consumers. In addition, dedicated domain data services provide the mean to persist data and access it still through the integration fabrics.

According to ZSM principles, management services can be logically grouped according to the functionality offered (such as data collection, analytics, intelligence, orchestration, control). Figure 7.5 depicts the ZSM framework reference architecture. Due to its native openness and architecture flexibility, the ZSM framework is able to integrate management services as implemented by Open Source framework or other SDOs, like for example the ETSI NFV MANO and its opensource implementation ETSI Open Source MANO (OSM) [35]. The possibility to flexibly compose management services, together with the exchange of management data provide the foundation of an innovative agile service management that makes easier the integration of the various management aspects (from data collection to orchestration and analysis) enabling the closure of the control loop through network and service optimization processes.

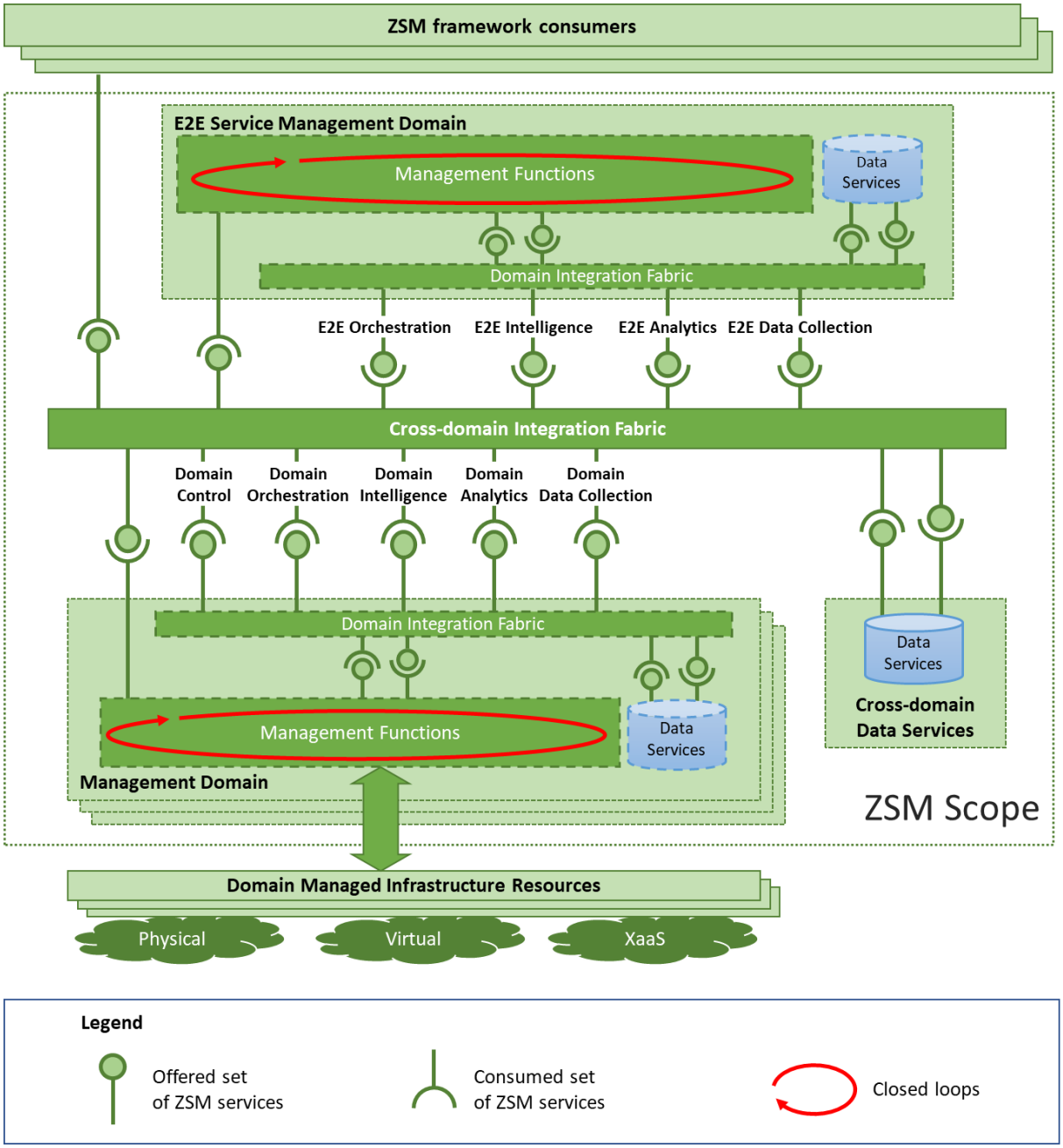


Figure 7.5 ETSI ZSM reference architecture (source: ETSI ZSM 002 [34])

Based on all of the above, the ETSI ZSM approach and principles fits with the LOCUS requirements and needs for what concern an agile framework for composing different localization services under a common agile and flexible architecture, where the integration fabric can drive the cooperation of orchestration, data collection and storage, analytics services following a Service Based Architecture (SBA) approach with decoupled services and functions. In practice, ETSI ZSM provides the means to enhance the ETSI NFV MANO capabilities (purely dedicated to orchestration and lifecycle management of standalone network services) with a more comprehensive approach where other crucial aspects of service operation is considered and tightly integrated, as data analytics and intelligence, which can

then be exposed to external entities to build applications on top. This is crucial for properly enabling the support of Smart Network Management and third-party and vertical use cases on top of the LOCUS localization services and functions. In practice, LOCUS is conceptually aligned to the ETSI ZSM architecture and principles, as data exposure and data analytics services as defined in ETSI ZSM 002 [34] are in line with the LOCUS capability to offer localization analytics and localization data as services to the application layer.

Moreover, ETSI ZSM takes intent based paradigm as one of its fundamental principles and means of automation, as a way to reduce the complexity of service offers exposed to external entities, with the aim of hiding the various technology and operation details (and constraints). With an intent based approach, services can be requested as simplified declarative policies or high-level service goals. This is also very relevant for the LOCUS localization as a service paradigm, where an intent based approach for exposure of data, analytics functions, and localization services in general, towards third parties and verticals can simplify the way such services are exposed and can be requested. This is mostly related on how LOCUS APIs are built, and in particular how LOCUS management and orchestration is able to translate intents and service goals into specific data collection and storage, analytics and machine learning constraints to compose the proper localization service on the fly.

8 LOCUS System Architecture and Deployment Options

This chapter describes the preliminary version of the LOCUS system architecture, taking into consideration the functional capabilities of the various LOCUS functions and applications described in the previous chapters. Therefore, it has to be considered as a first specification of the LOCUS system, to be consolidated and finalized in the context of deliverable D2.5, as part of the evolution of the technical work in WP3, WP4 and WP5, under the coordination of WP2.

The main idea behind this LOCUS system architecture definition is to facilitate the reuse of common localization and analytics functions for several purposes and applications scopes, thus enabling a flexible collection, manipulation, analysis and exposure of localization related data. In practice, LOCUS aims at providing a unified and generalized localization analytics platform that is able to support localization requirements from a wide plethora of applications, beyond those for Smart Network Management and vertical applications that will be developed within LOCUS.

8.1 LOCUS System Architecture

Figure 8.1 presents a comprehensive view of the LOCUS system architecture, and includes details of how the LOCUS functions presented and described in Section 4 are interconnected and deployed through the LOCUS management and orchestration functions on top of a virtualized infrastructure that spans across edge and core locations in the 5G network to implement the LOCUS services detailed in Section 6. Indeed, the LOCUS system is conceived to implement and expose localization analytics functions and services, tailored to localization requirements of Smart Network Management and Vertical applications, as described in section 6.

Following the 3GPP 5G Core architecture approach, LOCUS makes use of a Service Based Architecture (SBA), thus targeting lightweight interfaces across the LOCUS functions and enabling rapid interface development and high level of resource utilization. With the SBA approach, each LOCUS function provides an atomized capability, with high-cohesion, loose-coupling, and independent management from other functions. This allows each LOCUS functions to be deployed on-demand, updated and where needed decommissioned independently and with minimal impact on others. The LOCUS SBA approach envisages that each function produces certain outputs based on specific inputs, as expected by its service capability and required by the given Smart Network Management and Vertical applications specific needs. The LOCUS services, detailed in Section 6, are exposed through the Localization Analytics as a Service APIs as combination of multiple LOCUS functions atomized capabilities, that therefore need to be wired and linked together to provide the desired output requested by the LOCUS application.

This kind of chaining of LOCUS functions requires a workflow execution engine that is provided by the LOCUS Management and Orchestration, where a service lifecycle management capability is responsible to translate the service request into a number of functional steps that involve one or more LOCUS functions, as described in Section 4.4. Each step of a localization analytics workflow involves runtime discovery of the appropriate service endpoints to be invoked, dynamic preparation of the payload to be used and endpoint invocation. The result of each step can be also subject to be processed in the next ones. Beyond these function to function wiring capabilities, LOCUS enables interactions with the data persistency entities as well, as special service endpoints cases. Therefore, the data management service endpoints (for traditional query and storage of data into the data store, including the security and privacy functionalities) can be considered as part of the localization analytics workflows, supporting LOCUS functions that interact through the data they read and write in the data store persistency entities.

The output of each localization analytics workflow is then exposed to the given LOCUS application as service response through the Localization Analytics as a Service APIs. As described in Section 6, this can happen in three different ways: i) exposing CRUD operations on collection of structured and unstructured data persisted within LOCUS, ii) exposing the output as an on-demand REST service, iii) exposing the output as a continuous stream of data over a message bus topic (e.g. based on Kafka). This is done by the Localization Analytics Gateway depicted in Figure 8.1, which is responsible to manage and control the service response exposure towards the LOCUS applications. In addition, the gateway allows the LOCUS applications to inject data themselves into the LOCUS platform, in case they need to exploit it in their application logic, by regulating the access to the data persistency functions through the Localization Analytics as Service APIs.

In summary, the overall LOCUS SBA approach has the main goal to enable a flexible and composable platform where the various LOCUS functions can be loosely coupled and integrated while facilitating sharing and re-use of some of the key functionalities (e.g. those for the localization enablers or data security and privacy) across different LOCUS services and applications. Such approach also enables compatibility and integration of the LOCUS functions with the 5G Core components themselves, as depicted in Figure 8.1, combining the two SBAs under the same control and management umbrella and possibly using a common Service Based Interface. This provides more value to the LOCUS approach due to its alignment and compatibility with the 5G Core Architecture. In this respect, some of the LOCUS functions for the localization enablers and analytics may make use of 3GPP standard procedures to retrieve positioning data following the architecture described in section 7.1.

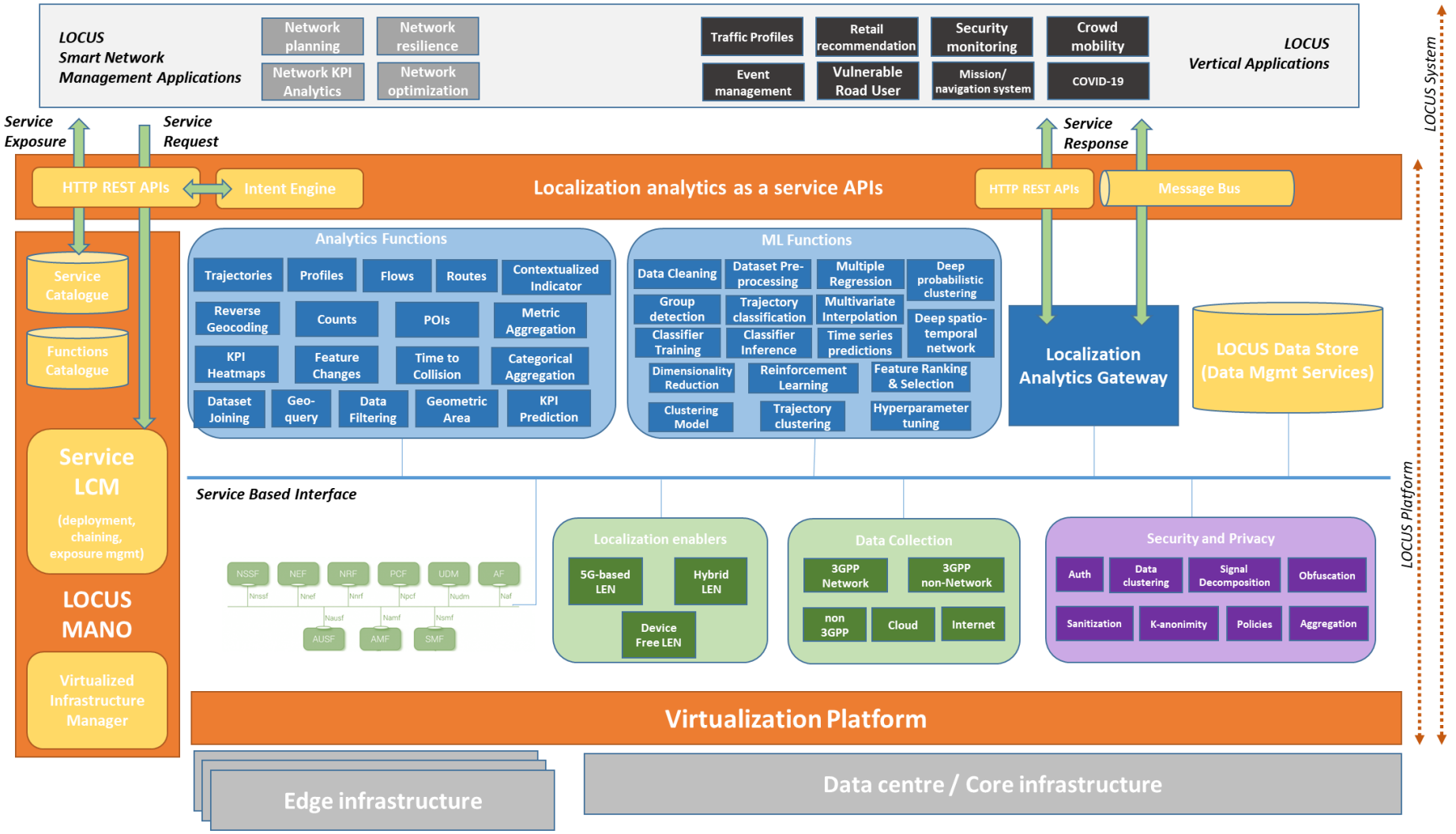


Figure 8.1 LOCUS System architecture

It is relevant to highlight that, with respect to Figure 8.1, the LOCUS platform is composed by the combination of the LOCUS MANO, the Localization Analytics as a Service APIs (including the gateway functionalities), the various LOCUS functions (analytics, ML, localization enablers, data collection, security and privacy), the virtualization platform for hosting and running such functions, while the whole LOCUS system comprises the Smart Network Management and Vertical applications as well. As anticipated above, with its unified and generalized approach, LOCUS facilitates other applications to leverage on the localization analytics services exposed by the Localization Analytics as a Service APIs.

The following sub-sections provide details about the LOCUS system capabilities, with focus on those platform components and functions not already described in section 4 and 5.

8.1.1 Virtualization platform

The virtualization platform is a technological pillar in the LOCUS system, as it enables to deploy and operate the various LOCUS functions described in Section 4 as virtualized functions, thus facilitating the provisioning of localization analytics services in a flexible, scalable and customisable way.

Indeed, LOCUS implements a full localization and analytics system sitting on top of the 5G network infrastructure, thus spanning edge and core data centre locations. While LOCUS concentrates on enabling localization functions and services on top of the 5G infrastructure, its scope and target in terms of service offer is transversal to the pure 5G connectivity. LOCUS enables operators and service providers to expose localization analytics as a service, leveraging 5G network information combined with a set of non-3GPP and external sources of positioning related data, as described in Section 4. The virtualization platform leverages on the NFV principles for implementing the LOCUS functions in software and run them on commodity infrastructures, therefore following the same model applied by operators and service providers to implement the traditional 5G connectivity services and 5G vertical use cases. In practical terms, the LOCUS virtualization platform allows to consider LOCUS as an augmentation of the 5G network management capabilities, offering to operators and service providers the possibility to exploit the localization related data and functions in two main directions: i) implement Smart Network Management applications; and ii) offer new location and analytics-based services to third parties.

The LOCUS virtualization platform provides an abstraction of the edge and core clouds in the 5G infrastructure, and exposes a virtualized infrastructure where the LOCUS functions can be dynamically and on-demand deployed to satisfy the requirements of the LOCUS applications. Therefore, LOCUS assumes to have access to a virtualized infrastructure where the various LOCUS functions can run as either microservices (i.e. implemented as Docker containers) or traditional Virtualized Network Functions (i.e. implemented as Virtual Machines), according

to the specific function requirements (e.g. in terms of size, time for spawning, scaling rules) on the one hand, and to the LOCUS applications constraints on the other.

In general, the LOCUS virtualization platform is under the management of the LOCUS MANO, and expose the 5G edge and core infrastructure as a set of computing locations and distributed clouds where the virtualized LOCUS functions can be automatically deployed and operated, as depicted in Figure 8.2. The selection of proper location (i.e. edge or core) where to spawn and run a specific LOCUS function also depends on the specific function requirements as well as on the constraints posed by the given LOCUS application for which such a function is required to be executed and invoked. In practice, specific localization enablers, or analytics and ML functions, may need to be deployed at edge cloud locations due to specific latency or user proximity constraints (e.g. in the case of LOCUS applications targeting Automated Guided Vehicles or Vulnerable Road Users).

In terms of virtualization platform technologies, LOCUS will make use of de-facto standard technologies for the management of the virtualized infrastructures, such as Openstack [20] and Kubernetes [21].

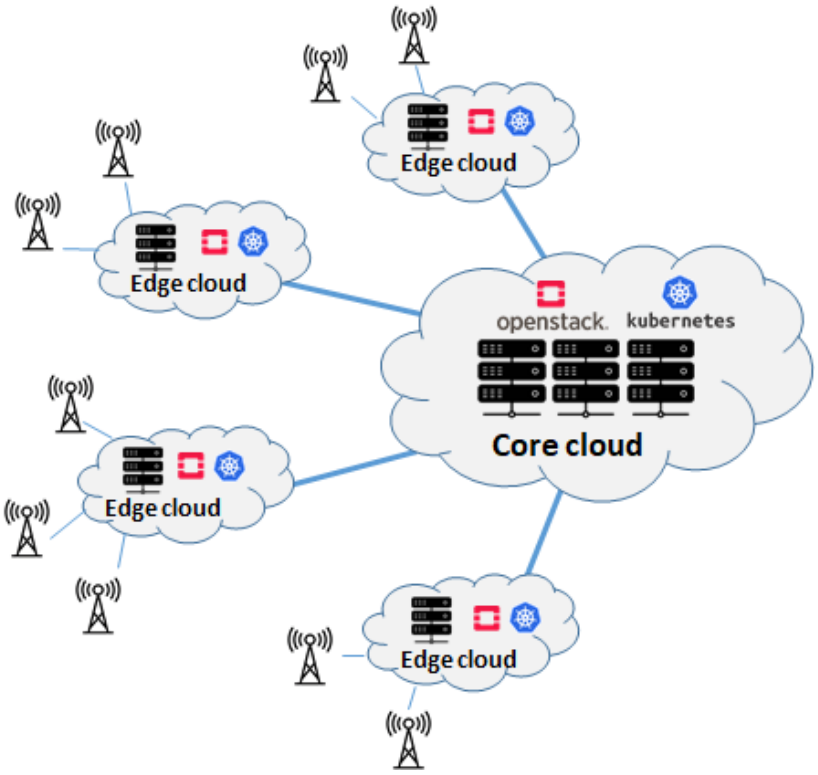


Figure 8.2 LOCUS virtualized infrastructure

8.1.2 LOCUS Management and Network Orchestration

The LOCUS MANO allows managing the lifecycle of the LOCUS functions in an automated and dynamic way, following the requirements of the various Smart Network Management and third-party vertical applications. Beyond the alignment with the ETSI NFV principles for

functions virtualization and their management, the localization services orchestration in LOCUS is also taking a step forward with respect to the current monolithic service management solutions, mostly dedicated to the management of NFV-like pipelined services, where the lack of agility in the service lifecycle and operation is a clear limitation. Indeed, LOCUS follows the principles of zero-touch service management defined in ETSI ZSM, that targets a novel, horizontal and vertical agile management and automation of emerging services, including 5G ones. In practice, following the ETSI ZSM architecture capabilities (as described in section 7.3), the LOCUS MANO is also responsible for the exposure of localization data and analytics produced and stored within the LOCUS platform as services. In this scenario, the LOCUS MANO layer manages the automated deployment, configuration, and operation of the heterogeneous localization and analytics functions, including if required their integration with the 5G Core components, by coordinating the selection of edge and core infrastructures for optimal operation. The LOCUS MANO is therefore the primary manager of the virtualized infrastructure, and practically translate the LOCUS service and application requirements expressed through the Localization Analytics as a Service APIs into LOCUS functions to be dynamically and on-demand instantiated, configured and wired together to provide the required localization analytics features. It enables a multi-tenant approach, at both the MANO and virtualized edge and core infrastructure layers, by ensuring well-separated and isolated running environments for localization enablers and functions (including related data) serving applications from different owners, being them either Smart Network Management or Vertical related ones. Such a multi-tenant approach allows to separate the management and orchestration of the various LOCUS services, thus having independent lifecycle and avoid interferences among them. However, given the SBA nature of the LOCUS approach, the MANO is also responsible to manage and control the re-use and share of LOCUS function instances across different LOCUS services and applications, e.g. when they are related to the same owner.

As said, the LOCUS MANO is aligned the ETSI NFV MANO approach, and therefore provides a set of core functionalities for the management of the LOCUS functions in the form of virtualized functions. These functionalities are highlighted as functional boxes in Figure 8.1, and are detailed in the following table.

Table 8.1 LOCUS MANO Functions

LOCUS MANO Function	**Description**
Service Catalogue	It is the repository of the available localization services that can be instantiated by the LOCUS MANO in the virtualized platform. Each service is described following a common template and structure aligned with the ETSI NFV Network Service

	Descriptor (NSD), thus including information on the constituent LOCUS functions and how they are linked together (in terms of relation of their input/output data). Complex LOCUS services can be described by the combination of multiple NSDs, enabling re-use, sharing and composition of basic services.
Functions Catalogue	It is the repository of the available localization, analytics, ML, data collection, security and privacy virtualized functions. Each of these functions can be instantiated on-demand as part of a LOCUS service. The LOCUS functions in this catalogue are described following a common template and structure aligned with the ETSI NFV VNF Descriptor (VNFD) model, that provide details on virtualized resource requirements.
Service Lifecycle Management	It is the core engine of the LOCUS MANO, as it orchestrates the deployment and configuration of the LOCUS functions, taking care to wire them to satisfy the LOCUS application needs. In practice it translates the requirements from LOCUS applications, issued through the Localization Analytics as a Service APIs into specific localization, analytics, ML, security and privacy virtualized functions to be either re-used or newly instantiated. The Service LCM also takes care to optimize the placement of the various LOCUS functions involved in a given service request, selecting proper edge and core cloud locations according to their constraints in terms of required input data and localization scope.
Virtualized Infrastructure Manager	It is the LOCUS MANO function that manages the virtualized infrastructure and offer to the Service Lifecycle Management a common entry point for deploying the LOCUS functions as virtualized functions. It takes care to glue together virtualization and networking aspects from hybrid technology domains, such as those provided by Openstack [20] for Virtual Machines management and Kubernetes [21] for container orchestration.

8.1.3 Localization analytics as a Service APIs

The Localization analytics as a Service APIs represent the front-end of the LOCUS platform, and expose the LOCUS localization services towards the application layer. The main idea

behind these APIs is to provide a flexible and open interaction with the localization, analytics and ML functions hosted in the LOCUS platform, and enable the LOCUS applications for Smart Network Management and Vertical use cases to consume localization analytics on-demand and based on their application needs, in line with ETSI ZSM principles for data and analytics services exposure.

As anticipated in section 6, the LOCUS services exposed by the Localization Analytics as a Service APIs can be grouped in three main categories, according to the type of interaction implemented with the LOCUS applications:

- HTTP REST based access to localization related data stored in the LOCUS platform in the persistency entities
- HTTP REST based access to localization analytics and data as produced by the LOCUS localization, analytics and ML functions
- Message bus base access to access to localization analytics and data as produced by the LOCUS localization, analytics and ML functions

Therefore, the Localization Analytics as a Service APIs provide two means through which access and consume the localization analytics and data generated within the LOCUS platform:

- HTTP REST services, with traditional CRUD operations exposed toward the LOCUS applications
- Message Bus services, to subscribe and consume streams of localization analytics data, for example based on Kafka

However, to maintain control on authorization and authentication of LOCUS applications accessing the LOCUS platform, each of the service request/response transaction occurring through the APIs is by definition asynchronous and split into three separate phases:

- *Service request*: a LOCUS application issues a service request through a HTTP REST operation, requesting for the given localization analytics or data
- *Synchronous service response*: a service consumption endpoint is provided by the LOCUS platform, with two options: i) an HTTP REST service endpoint for the LOCUS application to access and retrieve the required localization analytics information, ii) or a message bus endpoint and topic to subscribe and start consuming continuous streams of data
- *Asynchronous service consumption*: the LOCUS application accesses the service consumption endpoint provided in the synchronous service response and consume localization analytics or data service requested

The access to localization analytics and localization data in general, as part of the service response phase and involving data and information produced by the various LOCUS functions is mediated and regulated by the Localization Analytics Gateway, which takes care, for each specific service request to guarantee that the involved LOCUS Smart Network Management

or Vertical application access and consume only the required data in a controlled and secure way. If needed, the LOCUS applications can also use the Localization Analytics as a Service APIs (with regulated access to the data store management services) to store the data they produce within the LOCUS platform, e.g. to make it available to other applications or to exploit it in their own application logics.

LOCUS enables two different approaches for the exposure of localization analytics services towards the application layer (as described in the two subsections below):

- Pre-defined localization analytics service exposure
- Intent-based localization analytics service exposure

While the two options are complementary, the pre-defined localization analytics services approach is considered essential and mandatory for the LOCUS platform implementation. On the other hand, the intent-based localization analytics services approach can be considered as a nice to have feature whose implementation will be evaluated in the technical WP activities (WP4, WP5 mostly) during the project, also taking into consideration the requirements from the LOCUS applications.

8.1.3.1 Pre-defined localization analytics services

The LOCUS localization services that are offered by the platform to the LOCUS applications can be seen as the composition of one or more LOCUS functions that need to be wired and chained together according to their input/output data requirements to achieve to achieve a specific analytics goal.

Indeed, a specific LOCUS application localization related requirement or need can be translated into the deployment of one or more LOCUS functions for collecting data, providing localization of users and devices, analysing it and applying ML to predict some specific behaviour or localization characteristic. In this case of pre-defined localization analytics services, such combinations of LOCUS functions can be defined in advance and properly modelled to instruct the LOCUS MANO for their automated deployment, when a given service is request by a LOCUS application.

Therefore, such service offers are stored in the LOCUS MANO Service Catalogue, where multiple combinations of LOCUS functions (modelled as individual VNFs) are exposed through the Localization Analytics as a Service APIs in the form of NSDs.

In practice, the exposure of these pre-defined service offers is made through a set of HTTP REST APIs that, as depicted in Figure 8.1, provide access to the service descriptions in the Service Catalogue. Therefore, the pre-defined localization analytics services management workflow, as supported by the LOCUS platform and enabled by the Localization Analytics as a Service APIs (in accordance with the asynchronous service request/response transactions defined above) can be summarized in the following steps:

1. *Service exposure*: a LOCUS application queries the MANO Service Catalogue, and retrieves the list of localization analytics services that can be activated on-demand.
2. *Service request*: after having selected the localization analytics service of interest, the LOCUS application issue a request of service activation, specifying its constraints (at least in terms of desired service response access format, i.e. HTTP REST service vs. message bus continuous data stream)
3. *Service preparation*: the LOCUS MANO Service LCM validates the service request, performs some feasibility check (in terms of available resources and required localization related data and information), identifies the LOCUS functions to be either instantiated or re-used, prepares the service consumption endpoint (in the form of HTTP REST or message bus service according to the request), that is configured and managed on the Localization Analytics Gateway, and trigger the activation and on-demand instantiation of the various LOCUS functions
4. *Service response*: in the synchronous service response, the LOCUS MANO Service LCM provides to the given LOCUS application a service identifier (to be used for service status queries and subsequent operations, e.g. termination) and the details of the service consumption endpoint
5. *Service activation*: in this step, the Service LCM takes care to follow the service activation, creating all of the required LOCUS functions instances and configuring their interconnections in the virtualized infrastructure to properly realize the requested service according to the requirements in terms of input/output data for each function.
6. *Service consumption*: once the service is ready to be consumed, the LOCUS application can access the service consumption endpoint and start accessing the localization analytics data of interest, through the mediation of the Localization Analytics Gateway

This catalogue based way of exposing services towards the LOCUS application layer follows a traditional telco-oriented approach, also part of well-defined standard approaches in TM Forum (with the Information Framework – SID [36]) and ETSI NFV itself (with the Network Service and VNF Descriptors models) for example, and make the LOCUS platform aligned with related telco service management and orchestration solutions. In particular, this can be practically pursued by adopting de-facto standard existing network and service management open source tools as a baseline for the LOCUS MANO, like ETSI Open Source MANO [35], as detailed in section 8.2.3.

8.1.3.2 Intent-based localization analytics services

Complexity of network related services is continuously growing due to increasing number of technologies, network layers and dynamics, practically driven by virtualization. The request for fulfilment of a given service can become very complex and pose many different constraints depending on multiple factors and parameters.

Intent is a way to abstract service complexity at API level. An intent can be seen as a simplified service goal, expressed in natural language and linked to a declarative policy, and intent-based APIs allow the request for service intents. In turn, an intent-based system follows two main principles:

- Translation and validation: the system translates a service intent into concrete actions and operations that the system itself can perform to achieve the service intent goal. In parallel, it verifies that the requested intent can be satisfied according to the system capabilities.
- Automated implementation: when the service intent is translated and decomposed into actions and operations, these are automatically implemented and applied to deploy the actual service instance

In practice, following these principles, the LOCUS platform can provide an alternative mechanism (with respect to pre-defined services described above) to define and customize localization analytics services through the Localization Analytics as a Service APIs, adopting an approach based on service intents expressed in natural language. This is done by allowing LOCUS applications to request localization analytics service through a sentence in plain English that describe the service constraints and requirements. Such request can be issued through the Localization Analytics as a Service APIs, and then processed by the Intent Engine embedded in the API layer to apply translation and validation of the requested service intent.

In this case, the intent-based localization analytics services management workflow does not include an explicit service exposure phase, and can be summarized in the following steps:

1. *Service intent request*: the LOCUS application requests for a service intent issuing an HTTP REST request containing the intent sentence through the Localization Analytics as a Service APIs, and specifying the desired service response access format (i.e. HTTP REST service vs. message bus continuous data stream)
2. *Service intent translation*: the Intent Engine translates the service intent into localization analytics service constraints, looking for a service descriptor in the MANO Service Catalogue that could fulfil them. If not found, the Intent Engine can further look into available LOCUS functions in the Functions Catalogue and create a new ad-hoc service descriptor to serve the intent
3. *Service preparation:* the Intent Engine requests to the LOCUS MANO Service LCM to activate the given selected service. Similar to the pre-defined service case, the Service LCM identifies the LOCUS functions involved, prepares the service consumption endpoint and trigger the activation and on-demand instantiation of the various LOCUS functions
4. *Service response*: as in the pre-defined service case, the MANO Service LCM provides to the given LOCUS application a service identifier and the details of the service consumption endpoint

5. *Service activation*: as in the pre-defined service case, the Service LCM takes care to follow the service activation, creating all of the required LOCUS functions instances and wiring them in the virtualized infrastructure
6. *Service consumption*: once the service is ready to be consumed, the LOCUS application can access the service consumption endpoint and start accessing the localization analytics data

In summary, the LOCUS intent-based approach for the Localization Analytics as a Service APIs can simplify the process of service definition and customization, enabling an easy access to the localization analytics services offered by the LOCUS platform.

8.1.4 The Localization Analytics Gateway

The Localization Analytics Gateway is the main access point for the LOCUS applications to consume the localization analytics data generated by the services they request. As part of the asynchronous localization analytics service consumption offered by LOCUS to the application layer through the Localization Analytics as a Service API, the Localization Analytics Gateway guarantees that applications are given access to the only analytics data generated in the context of their services, thus providing data isolation across multiple services and applications. For this, it leverages on the LOCUS security and privacy functions to provide LOCUS applications secure, authorized and authenticated access to localization analytics data and stored in the persistency entities part of the data store.

Through the Localization Analytics Gateway, the LOCUS applications can also inject additional analytics data into the LOCUS platform that can be possibly exploited by other applications, or by the internal LOCUS functions themselves. This option is subject to be validated by the LOCUS platform in the phase of the service request, as a service consumption constraint posed by the application itself.

Different deployment models are supported in LOCUS for the Localization Analytics Gateway. As a special localization analytics and data exposure management function, it can be either deployed in single or multiple instances (e.g. one for each service consumed by a given LOCUS application as part of the on-demand service creation) within the LOCUS platform.

In any case, its configuration and behaviour are under the control of the MANO Service LCM, which according to the service constraints and characteristics expressed by LOCUS applications in the service request, instructs the Localization Analytics Gateway to apply a suitable analytics data exposure model. According to the two options available to consume the localization analytics data, the Localization Analytics Gateway supports both HTTP REST and message bus-based models, being in practice responsible for implementing the exposure services for the Localization Analytics as a Service APIs as depicted in Figure 8.1.

8.1.5 Data Management

Given the heterogeneity and diversity of data sources, and data types in general, that, as described in section 4.2.1, LOCUS needs to support, a data lake approach is required to properly manage the various data collected, but also produced and processed by the localization and analytics related functions described in section 4.2.

The LOCUS data store indeed can be seen as a logically centralized repository for heterogeneous structured (row and column) and unstructured non-tabular raw data in its native format. In practice if offers the possibility to store data in open formats (i.e. without the need to define or impose data structures or schemas) to enable and ease data analytics on top of it. As described in the previous sections, the LOCUS data store allows different data ingestion modes, including real-time, batch and data streams, and the various LOCUS functions can also use it to store their processed data, in either structured or unstructured formats. The various data security and privacy functions listed in section 4.2.4 are also highly linked to the data management functions offered by the data store.

In general, the LOCUS data store provides data manipulation, transformation and persistence functionalities that can be leveraged by all of the other LOCUS components and functions to access, persist and manage the data they require. A preliminary list of these data management functions is provided in the Table 8.2 below, and will be refined and augmented as required while LOCUS evolves in the final architecture deliverable D2.5.

Table 8.2 LOCUS data management functions

Functionality	**Type**	**Description**
Unstructured Data Persistence	Persistence	This function is used to persist object-based and document-based data (e.g. blob/serialized/text) into the LOCUS data store. The retrievability of this object will be based on the key semi-structured (JSON) meta-data provided.
Structured/Relational Data Persistence	Persistence	This function is used to persist multi-dimensional datasets of numerical and categorical features into the LOCUS data store. The retrievability of these data is based on SQL-compliant methods.
Unstructured Data Query	Query	This function is used to gain access to unstructured data that are stored in the data store. It is parametrized by the query definition on the unstructured dataset's index.

Structured Data Query	Query	This function is used to gain access to structured (tabular) data that are stored in the data store. It is parametrized in an SQL-compliant language/format query.
Structured Dataset Transformation to Unstructured or Semi-Structured	Transformation	This function refers to the transformation of multidimensional tabular-based datasets into semi-structured or unstructured forms to be used by a LOCUS localization or analytics function. The implementation of the mapping logic needs to be provided as input.
Unstructured Dataset Transformation to Structured	Transformation	This function refers to the transformation of an unstructured object into a multidimensional structured dataset. The mapping function, as well as the target dataset structure/format, are to be provided by the instantiation of this function.
Unstructured Dataset Transformation to Structured	Transformation	This function refers to the transformation of an unstructured object of a given type into another unstructured object. The mapping logic is to be provided as input to this transformation function.

8.2 Implementation options for the LOCUS architecture components and functions

This section provides few technical insights on implementation and deployment options for the LOCUS architecture components and functions. The listed technologies, solutions and tools are currently under detailed analysis in the various technical WPs (WP3, WP4, WP5), for proper selection of relevant solutions, under the coordination of WP2.

8.2.1 LOCUS functions

As described and detailed in the previous sections, one of the main innovative LOCUS proposals is the virtualization of the localization related functions, in the context of a flexible localization and analytics platform that can support multiple service requirements in the scope of Smart Network Management and Vertical applications. The idea is to not develop and produce ah-hoc localization and analytics functions for each external application, but rather implement a comprehensive set of localization related functionalities that can be virtualized, combined and chained together to support heterogeneous localization services and use cases. The goal is indeed to re-use the same functions (localization, analytics, ML, data collection,

security and privacy), either sharing them across several localization services and applications, or creating multiple isolated instances to separate their running environments.

When dealing with applications development and deployment in virtualized environments, two main options are available for their deployment (with some implications on their design and implementation as well) into virtualized platforms and infrastructures: monolithic (Virtual Machine based) architectures and microservice architectures.

8.2.1.1 Monolithic architectures

The monolithic architecture is considered a traditional way of designing and building applications. A monolithic application is normally made as a single and indivisible unit. All of the application functions are managed, hosted and deployed in a single running environment. A monolithic application can indeed include several functions, like different application or business logics, modules databases, external and internal interface, all of them unified and interconnected in a single, self-contained unit.

Such kind of applications are typically simple to develop when compared with microservice applications, and they are also easy to deploy as they are produced as a single executable file. These kinds of monolithic applications are well suited for being deployed in virtualized environments as Virtual Machines (VMs), thus as software-defined computers with their own operating system that run on a host server with a different underlying operating system. This makes easier the duplication and the provisioning of multiple instances of the same application in virtualized scenarios. However, depending on the size of the application (or the VM) the start-up time can drastically slow down, and also for each update (or customization) of any of the application module logic a full re-deployment of the entire application is needed. Moreover, monolithic applications can also be difficult and not efficient to scale when different modules have conflicting resource requirements.

In summary, in LOCUS, the use of monolithic architectures would generate a lack of flexibility in the realization of localization services, as it would require to build ad-hoc compositions and integration in the same application of different localization, analytics, ML and data management functions in support of specific Smart Network Management or Vertical use cases.

8.2.1.2 Microservice architectures

Microservices is an architectural approach for developing an application as a collection of small services, each implementing self-contained capabilities, running its own process and communicating via well-defined APIs and messaging channels. Microservices can be deployed, upgraded, scaled, and restarted independently in the application, typically as part of an

automated system, thus enabling runtime updates and upgrades without affecting overall offered services.

Following this principle, applications and services can be designed as a composition of microservices each providing an atomic function, including data collection and storage, specific application logic, OAM, monitoring and analytics. In addition, this approach allows dynamically plugging into running application instance additional functions in the form of new microservices to tailor the microservice application at runtime.

This model is also fully aligned with the cloud-native approach, which aims at building and running applications exploiting the advantages of the cloud computing delivery model, thus caring more about how applications are created and deployed, rather than where. In other words, developing an application following the microservices architecture makes it automatically cloud-native. In this context, developers aim at building and operating cloud-native applications and services that automate and integrate the concepts of microservices and containers. Indeed, containers offer a cloud-native way to deploy microservices based applications, as a more efficient, light, scalable and reactive approach compared to VMs based silo deployments.

In LOCUS, microservice architectures are suitable to be applied for the localization services in support of Smart Network Management and third-party and vertical applications, as well as for the management and orchestration platform itself. By following the microservices approach and adopting a service-based architecture, LOCUS enables the packaging of localization and analytics functions as cloud-native applications. This allows chaining the various LOCUS functions, if needed with the 5G Core components, to provide the required localization and analytics services.

8.2.2 Packaging of LOCUS functions into virtualized functions

The LOCUS platform is conceived to support localization, analytics and ML functions packaged in two main alternative forms: containers or Virtual Machines. This allows to implement a hybrid approach for the management and orchestration of any type of developed function in LOCUS, supporting more traditional telco-oriented IaaS virtualization approaches as well as microservices oriented lightweight packaging models and container orchestration solutions. This is aligned with the LOCUS virtualization platform solution that is defined as a hybrid virtualized infrastructure where Openstack [20] and Kubernetes [21] coexist.

In terms of LOCUS functions packaging options, a container is a standard unit of software that packages code and all its dependencies, virtualizing the operating system instead of the hardware, offering a highly portable and efficient solution. Docker is the de-facto container packaging standard, and provides lightweight, standalone, executable package of software

that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

On the other hand, Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows to virtualized the hardware for multiple VMs to run on a single machine, each with its own operating system. Each VM indeed, includes a full copy of an operating system, the application, and all of the necessary binaries and dependency libraries.

The LOCUS functions, independently from their nature (i.e. container vs VM), are packaged and modelled as VNFs. In this respect, ETSI NFV defines a VNF Package as a common and unified way to package and describe virtualized network functions to be then managed and orchestrated through MANO tools. Indeed, the ETSI VNF Package is considered as a standard and uniform way for virtualized function providers to deliver them as VNFs. ETSI NFV takes the TOSCA YAML Cloud Service ARchive (CSAR) definition as the reference for VNF packaging [37], where a CSAR file is mainly a .zip file with a well-defined structure containing all of the required VNF information required to manage its lifecycle, including deployment and runtime operation in virtualized infrastructures.

The main structure of a VNF package following the CSAR approach is the following:

- **Metadata information**: information about the name of the package, its version, its provider, the TOSCA and CSAR versions, etc. This information is included in a *TOSCA.meta* file which also links to the main VNF description included in the VNFD YAML file.
- **VNF Descriptor**: it is the main TOSCA definitions YAML file in which information for package onboarding and VNF management is provided, in terms of VNF properties and requirements [32]:
    - Virtualized resources required by the VNF (compute, storage, networking)
    - Connectivity specifications, such as virtual interfaces required by the VNF
    - Lifecycle management behaviour, runtime operations and configuration, including scaling and autoscaling policies.
    - References to software images to be used for running the VNF, lifecycle management scripts and other files contained in the VNF package.
    - Deployment flavours, defined as deployment options in terms of resource requirements and connectivity specification. The same VNF can be possibly created in different flavours.
- **Manifest file**: this file provides package integrity and authenticity. It contains some metadata about the VNF (name, provider, version, release date), and it also contains the URL for each artefact contained related to the VNF package. For security purposes,

a hash code could be added to each artefact specifying the algorithm used to generate this hash code.

- **Software images**: optionally, the VNF package can embed the actual software images (containers or VM images) required to run the VNF. Other option is to have them linked in the Manifest file.
- **Change History file**: this file is a human readable text file where all changes made on VNF package are versioned, tracked and inventoried.
- **Testing files directory**: testing files to enable the validation of the VNF package. The provider can include other files in this directory, containing additional information, such as the test descriptions.
- **Licensing information directory**: it provides the license term for the whole VNF as well as other license terms for other artefacts included in the package if their license is different from the VNF one.
- **Certificate files**: they aim at adding security to the VNF package. With these certificate files, the VNF provider can digitally sign some of the artefacts inside the VNF package.
- **Additional files**: any additional file linked in the VNFD for specific management scopes, like lifecycle management scripts, vendor-specific files, etc.

8.2.3 LOCUS MANO

8.2.3.1 Open source tools

8.2.3.1.1 ETSI OSM

Open Source MANO (OSM) [35] is an operator-led ETSI community that is delivering a production-quality open source Management and Orchestration (MANO) stack aligned with ETSI NFV Information Models and that targets to meet the requirements of production NFV networks. OSM has the main objective of being a world-class production ready solution for NFV management and orchestration, and it is engineered, tested and documented to be functionally complete to support to be a key component for internal/lab and external/field trials as well as interoperability and scalability tests for VNFs and Network Services. It allows for rapid installation in VNF vendor, system integrator and operator environments. OSM provides a full solution for NFV management and orchestration, covering both design time (i.e. DevOps) and run time phases of VNFs and Network Services lifecycle management. OSM logical architecture including the functionality offered by OSM is shown in Figure 8.3, where different colours identify run-time and design time components. The design-time scope of OSM includes: i) the capability for Create/Read/Update/Delete (CRUD) operations on the Network Service definition, ii) tools for VNF Package Generation, iii) Graphical User Interface

(GUI) to accelerate the design time phase, VNFs on-boarding and deployment. On the other hand, the run-time scope of OSM mostly provides: i) an automated Service Orchestration environment as the main coordination engine, ii) plugins for integrating multiple Virtual Infrastructure Managers (including OpenStack, Kubernetes, VMWare, Amazon Web Services, OpenVIM), iii) plugins for integrating multiple SDN controllers, v) support to integrate Physical Network Functions into Network Services.

OSM provides a well-known, complete and unified information model to facilitate an accurate and sufficient description of the internal topology, procedures and lifecycle of Network Services. This information model is aligned with the ETSI NFV specifications and is at the base of the OSM lifecycle management procedures for Network Services and VNFs. On top of this model, OSM expose a set of northbound APIs aligned with ETSI NFV SOL005 specifications, with the addition of Network Service runtime operation and re-configuration through service primitives.

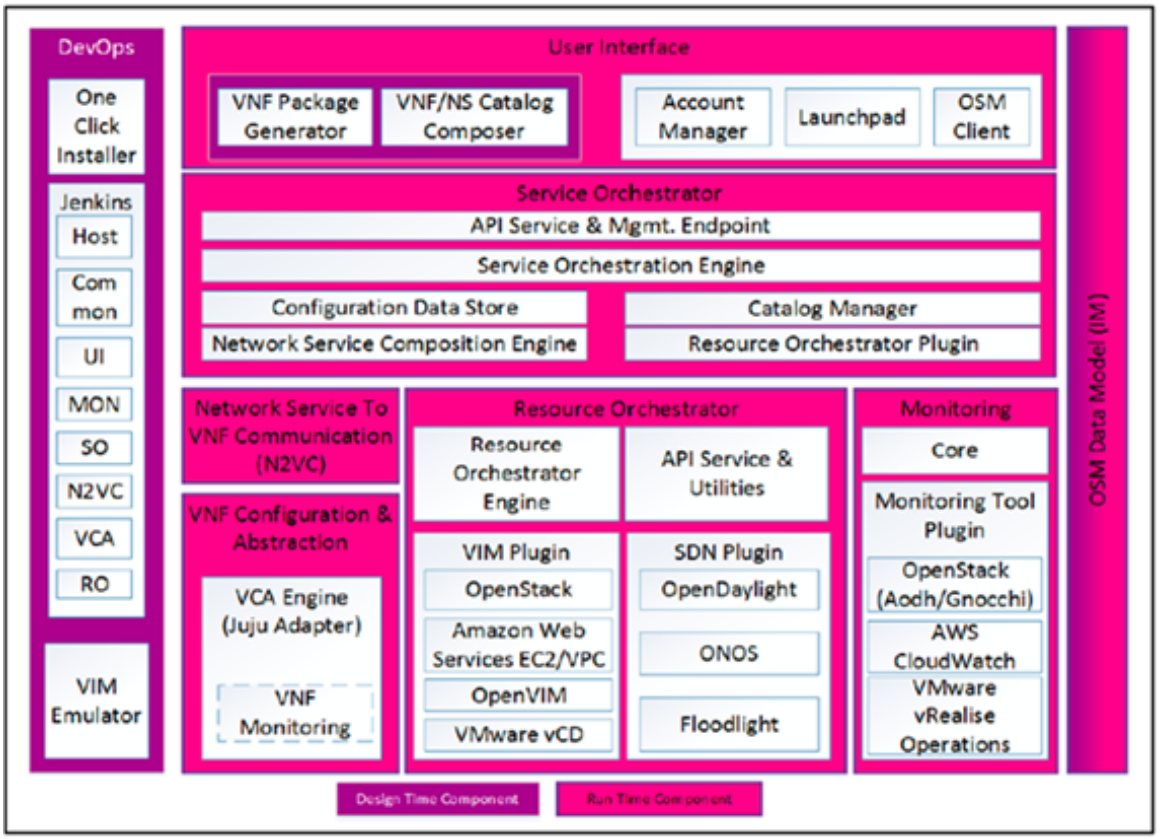


Figure 8.3 OSM logical architecture (source: ETSI OSM wiki, https://osm.etsi.org/wikipub/index.php/Welcome_to_OSM!)

The core functionality offered by OSM is the on-demand provisioning of Network Services over virtualized infrastructures.

A Networks Service is therefore the atomic logical service unit that can be invoked on OSM through its northbound APIs. Upon a Network Service creation request, OSM decomposes the Network Service into constituent VNFs to be instantiated and chained together to realize the

required service, following the definitions and constraints defined in the NSD, taking care to orchestrate their deployment and configuration in the virtualized infrastructure according to the requirements expressed in each VNF Descriptor. Therefore, the NSD provides a parametrized instantiation template that can be customized with specific service constraints and attributes for each given instance of a Network Service created from the same NSD. With respect to ETSI NFV specifications, the Network Service model is enhanced in support of, beyond the traditional telco oriented VNFs, Containerized Network Functions (CNFs) to be deployed on top of Kubernetes clusters. This approach allows a distributed edge/core deployment of Network Services over end-to-end virtualized infrastructures, integrating different virtualization approaches (i.e. traditional IaaS and containers) in support of hybrid services and applications. Moreover, as a further extension, OSM allows to operate at runtime directly on Network Service and VNF instances for so called "Day-2" configurations, that can directly and automatically update the service and application behaviours and structure, including scaling of individual VNFs.

In practice, as said, OSM offers a comprehensive support for the lifecycle management of Network Services and VNFs, covering all of the phases from service design, to package onboarding, up to automation of service creation and runtime operations. Table 8.3 below lists the main functionalities provided by OSM for each of these phases covering the Network Service lifecycle.

Table 8.3 OSM functionalities in the different Network Service lifecycle phases

Lifecycle Phase	Functionality	Description
Design	Network Service Package Definition VNF Package Definition	A two-level definition of service and function packages enables the re-use and share of same functions across several services The NS Package defines components, topology, supported operations, runtime primitives and the overall behaviour of a Network Service. The VNF Package defines the internal function topology, the required resources to run it and the runtime lifecycle and configuration operations allowed (e.g. scaling policies).
On-boarding	Create, Read, Update, Delete (CRUD) on NS Package and VNF Package.	OSM exposes CRUD operations on both NS Packages and VNF Packages to upload new service and functions descriptions. These are validated to check consistency among each of the service and function components and entities.

Network Service Creation	New Network Service instance creation.	The Network Service creation request triggers the deployment of a new instance of the service in the virtualized infrastructure, according to requirements and constraints expressed in the NS Package, combined with additional configuration and operation parameters issued in the request (e.g. for selection of proper functions placement).
Network Service runtime operation	Network Service instance lifecycle operations	OSM exposes primitives to request for "manual" execution lifecycle actions on Network Service instances, e.g. scaling, start/stop/resume, software upgrade, monitoring data collection.
	Network Service configuration primitives	Beyond regular lifecycle operations, OSM allows to define additional configuration primitives within the NS and VNF Packages defining internal execution commands (e.g. at VNF level), to be called at runtime through dedicated REST APIs and configuration parameters.
	Automated Network Service operation	OSM supports automated execution of lifecycle operations, without explicit trigger from the northbound APIs. These are driven by policies that can be defined at both NS and VNF level in the packages. These automated operations are normally applied based on monitored metrics, at both infrastructure and application level, e.g. to regulate automated VNF scaling operations.
Network Service Termination	Network Service instance deletion	This operation triggers the termination of a Network Service instance that in turn deletes all of the related VNFs and de-allocate the virtualized resources in use for them.

8.2.4 Chaining of LOCUS functions to achieve complete localization analytics services

As already described in the previous sections, the LOCUS platform enables the implementation of localization analytics services as the combination of one or more LOCUS functions that according to their input/output data requirements have to be wired and interconnected following a specific service topology.

For applying such functions interconnection different options are available, especially when dealing with functions virtualization and virtualized infrastructures like in LOCUS, where the various functions are implemented as Virtual Machines (e.g. running in Openstack infrastructures) or microservices (e.g. running as containers in Kubernetes clusters). LOCUS

envisages to consider two possible approaches: the ETSI NFV VNF Forwarding Graph (VNFFG) for traditional VNF based LOCUS functions, and Network Service Meshes for microservices and container-based functions.

The ETSI NFV VNFFG allows to define how VNFs are interconnected and wired at the traffic level [33]. In practice it specifies a service forwarding path and describes how the traffic should flow across the various VNFs in a given service, in terms of virtual interfaces (called in the VNF Descriptors as connection points) ordering. The VNFFG description is included within the Network Service Descriptor and makes reference to connection points defined in each constituent VNF Descriptor. In LOCUS, the VNFFG can be used to model how the localization, analytics, ML and other functions have to be interconnected to satisfy their input/output data constraints for providing the required localization analytics service. VNFFGs may be also created or updated on-the-fly (e.g. in the case of intent-based localization analytics service approach) according to changing or new input/output data requirements for LOCUS functions interconnections. Currently, VNFFGs can be managed and enforced through ETSI OSM [38], by leveraging on dedicated Openstack [20] networking capabilities for service function chaining [39].

On the other hand, Network Service Mesh (NSM) is a novel approach for providing and simplifying cloud-native enabled connectivity among workloads, such as containers, in Kubernetes [21]. NSM is inspired by the Service Mesh approach for creating dedicated infrastructure layers for facilitating service-to-service communications between microservices. While Service Meshes work at L4 and above, NSM targets L2/L3 connectivity among microservices deployed in Kubernetes. Currently available microservices and container networking technologies for Kubernetes mostly focus on homogenous, low-latency, high-throughput, immutably deployed application clusters. These assumptions do not match the needs of emerging telcos and enterprise services in highly dynamic and scalable 5G virtualized infrastructures. NSM proposes a new way to realize highly dynamic and on-demand connectivity services among workloads (e.g. containers, PODs). It defines an NSM Network Service as a dynamic and composable service built of smaller, reusable and scalable endpoints. In NSM and endpoint can be any workload, like a container, a POD, a virtual machine, or even an external client. NSM realizes these connections as point to point communication paths between the endpoints within a Network Service, and manages the lifecycle of the connections to create, re-route and destroy them during the lifetime of the NSM Network Service. In practice, NSM provides a simple set of APIs designed to facilitate connectivity either between containers running services or with an external endpoint, and deploys an NSM Manager on each node in the Kubernetes cluster that act as control plane for handling service discovery, service routing and service connection management to create the virtual wires among workloads.

With respect to the VNFFG approach, NSM implements a highly and natively dynamic connectivity paradigm to wire microservices running as containers in Kubernetes, that can be leveraged in LOCUS for chaining and interconnecting the various localization, analytics and ML functions to realize localization analytics services to be exposed to the application layer.

9 Conclusions – Next Steps

LOCUS aims at providing a unified and generalized localization analytics platform that is able to support a wide variety of applications defined within LOCUS project.

This deliverable elaborates on the reference architecture of LOCUS taking into account: (a) the LOCUS UCs as defined in Deliverable 2.1 [1] and their high-level requirements to be supported by the LOCUS Platform; (b) the security and privacy requirements that are related to UCs; and (c) the system-level requirements.

Following a short review of similar architectures, the deliverable presents the main functional blocks and then goes into more detail, defining the various functions expected within the LOCUS platform, indicative App descriptions for the suggested UCs, and services to be exposed by the LOCUS platform.

A preliminary, comprehensive overview of the LOCUS system architecture including how the various functions are placed and interconnected within the platform was also presented. The deployment aspects through the LOCUS management and orchestration functions on top of a virtualized infrastructure spanning across edge and core locations have also been described. Furthermore, the architecture described in this deliverable is aligned with the work being carried out in the various WPs and defines the specifications with respect to which the actual implementation takes and will continue to take place.

Lastly, while this deliverable is a preliminary version and may be subject to changes and additions, the main pillars for the architecture definition have been set and will be further detailed in the next months as the project progresses. The final version of this work will be included in Deliverable 2.5.

References

[1] Deliverable 2.1 EU LOCUS project “Scenarios, use cases and requirements”

[2] R. Di Taranto, L. S. Muppirisetty, R. Raulefs, D. Slock, T. Svensson and H. Wymeersch, "Location-aware communications for 5G networks", IEEE Signal Processing Mag., vol. 31, no. 6, pp. 102-112, Nov. 2014.

[3] 3GPP Technical report, TR 23.731, “Study on enhancement to the 5GC LoCation Services (LCS)”, December, 2018

[4] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements," in IEEE Signal Processing Magazine, vol. 22, no. 4, pp. 41-53, July 2005.

[5] J. Trogh, D. Plets, E. Surewaard et al., “Outdoor location tracking of mobile devices in cellular networks”, J Wireless Com Network, 115 (2019)

[6] BeFEMTO project consortium, “BeFEMTO: Broadband Evolved FEMTO Networks”, http://www.ict-befemto.eu

[7] S. Fortes, A. Aguilar-García, R. Barco, F. B. Barba, J. A. Fernández-luque and A. Fernández-Durán, "Management architecture for location-aware self-organizing LTE/LTE-a small cell networks," in IEEE Communications Magazine, vol. 53, no. 1, pp. 294-302, January 2015.

[8] S. Fortes, “Context-Aware Self-Healing for Small Cell Networks” PhD. thesis, IC, University of Málaga 2017. https://riuma.uma.es/xmlui/handle/10630/15782

[9] S. Fortes, D. Palacios, I. Serrano and R. Barco, "Applying Social Event Data for the Management of Cellular Networks," in IEEE Communications Magazine, vol. 56, no. 11, pp. 36-43, November 2018, doi: 10.1109/MCOM.2018.1700580.

[10] HORIZON 2020 Programme, “High precision positioning for cooperative ITS applications”, http://www.hights.eu.

[11] M. Z. Win, F. Meyer, Z. Liu, W. Dai, S. Bartoletti, and A. Conti, “Efficient Multi-Sensor Localization for the Internet-of-Things,” IEEE Signal Process. Mag., vol. 35, iss. 5, pp. 153-167, 2018.

[12] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, “Soft Information for Localization-of-Things,” Proc. IEEE, vol. 107, iss. 11, pp. 2240-2264, 2019

[13] FP7 Programme, “EUWB - Coexisting Short Range Radio by Advanced Ultra-Wideband Radio Technology”, https://cordis.europa.eu/project/id/215669

[14] S. Bartoletti, W. Dai, A. Conti, and M. Z. Win, “A Mathematical Model for Wideband Ranging,” IEEE J. Sel. Topics Signal Process., vol. 9, iss. 2, pp. 216-228, 2015.

[15] D. Dardari, A. Conti, U. J. Ferner, A. Giorgetti, and M. Z. Win, “Ranging with Ultrawide Bandwidth Signals in Multipath Environments,” Proc. IEEE, vol. 97, iss. 2, pp. 404-426, 2009.

[16] FP7 Programme, “SELECT: Smart Efficient Location, idEntification and Cooperation Techniques”, https://cordis.europa.eu/project/id/257544/it.

[17] HORIZON 2020 Programme, “European Location As A Service Targeting International Commerce”, https://cordis.europa.eu/project/id/641526/it

[18] HORIZON 2020 Programme, “GNSS driven EO and Verifiable Image and Sensor Integration for mission-critical Operational Networks”, https://www.gsa.europa.eu/gnss-driven-eo-and-verifiable-image-and-sensor-integration-mission-critical-operational-networks.

[19] HORIZON 2002 Programme, “MOBNET MOBile NETwork for people's location in natural and man-made disasters”, https://cordis.europa.eu/project/id/687338

[20] Openstack, https://www.openstack.org/

[21] Kubernetes, https://kubernetes.io/

[22] 3GPP Technical Specification, TS 29.572, 5G System; Location Management Services; Stage 3, July 2019

[23] Apache Kafka, http://kafka.apache.org/

[24] ETSI TS 103 300-2 Intelligent Transport System (ITS); Vulnerable Road Users (VRU) awareness; Part 2: Functional Architecture and Requirements definition; Release 2, V2.1.1 (2020-05)

[25] 3GPP Technical specification, TS 33.107, 3G security; Lawful interception architecture and functions, Release 15, June 2019

[26] J. Mirkovic, “Privacy-Safe Network Trace Sharing via Secure Queries,” in Proceedings of ACM CCS Workshop on Network Data Anonymization, October 2008.

[27] V. Sharma, G. Bartlett, and J. Mirkovic, “Critter: Content-Rich Traffic Trace Repository”, In Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security (WISCS ’14). Association for Computing Machinery, New York, NY, USA, 13–20, 2014 DOI:10.1145/2663876.2663886

[28] 3GPP Technical Specification, TS 23.273, “5G System (5GS) Location Services (LCS) – Stage 2”, version 16.2.0, December 2019

[29] 3GPP Technical Report, TR 22.872, “Study on positioning use cases”, version 16.1.0, September 2018

[30] 3GPP Technical Report, TR 22.836, “Study on asset tracking use cases”, version 17.1.0, December 2019

[31] 3GPP Technical Specification, TS 22.261, “Service requirements for the 5G system”, version 17.3.0, July 2020

[32] ETSI GS NFV-IFA 011, “Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; VNF Descriptor and Packaging Specification”, https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/011/03.04.01_60/gs_NFV-IFA011v030401p.pdf

[33] ETSI GS NFV-IFA 014, Network Functions Virtualisation (NFV) Release 3;. Management and Orchestration; Network Service Templates Specification V3.3.1, September 2019, available at: https://docbox.etsi.org/isg/nfv/open/Publications_pdf/Specs-Reports/NFV-IFA%20014v3.3.1%20-%20GS%20-%20Network%20Service%20Templates%20Spec.pdf

[34] ETSI GS ZSM 002 v1.1.1, "Zero-touch network and Service Management (ZSM); Reference Architecture", https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf

[35] ETSI Open Source MANO, https://osm.etsi.org/

[36] Information Framework (SID), TM Forum, https://www.tmforum.org/information-framework-sid/

[37] ETSI GS NFV-SOL 004, “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package and PNFD Archive specification”, https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/004/02.07.01_60/gs_NFV-SOL004v020701p.pdf

[38] Virtual Network Function Forwarding Graph Descriptor (VNFFGD) in OSM, available at: https://osm.etsi.org/wikipub/index.php/OSM_RO_VNFFG_implementation

[39] https://docs.openstack.org/networking-sfc/queens/