D5.1

Requirements for E-CORRIDOR Architecture

WP5 – E-CORRIDOR Platform: Requirements / Architecture / Implementation and integration

E-CORRIDOR

Edge enabled Privacy and Security Platform for Multi Modal Transport

Due date of deliverable: 30/11/2020
Actual submission date: 30/11/2020

30/11/2020

Version 3.0

Responsible partner: HPE
Editor: Mirko Manea
E-mail address: mirko.manea at hpe.com

Project co-funded by the European Union within the Horizon 2020 Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Authors: Claudio Caimi, Mirko Manea, Patrizia Ciampoli (HPE), Fabio Martinelli, Paolo Mori, Giampiero Costantino, Ilaria Matteucci, Giacomo Giorgi (CNR), Christian Plappert (FhG), Mohammed Ammara (CLEM), Stefano Sebastio (UTRC)

Approved by: Sandeep Gupta (CNR), Florian Fenzl, Christian Plappert, Roland Rieke (FhG)

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
0.1	16-Sep-2020	M. Manea, C. Caimi	HPE	Initial ToC
0.2	10-Nov-2020	M. Manea, C. Caimi	HPE	Added evaluation of requirements from AT and ISAC pilots
0.3	18-Nov-2020	M. Manea, P. Ciampoli, C. Plappert	HPE, FhG	Added contributions
0.4	19-Nov-2020	M. Manea, P. Ciampoli, M. Ammara	HPE, CLEM	Added tools sections Merged contribution from CLEM about S2C pilot requirements
0.5	23-Nov-2020	M. Manea, Stefano Sebastio, P. Mori, F. Martinelli	HPE, UTRC, CNR	Merged review from UTRC Added contribution to requirements to dev/test/prod environments Merged contributions from CLEM and CNR
2.0	23-Nov-2020	M. Manea	HPE	Available for internal review
3.0	30-Nov-2020	M. Manea, F. Martinelli	HPE, CNR	Released

Executive Summary

Deliverable D5.1 specifies the set of E-CORRIDOR framework requirements for multi-modal transport systems as resulted from the requirements elicitation of the three pilots outlined in D2.1 (Airport and integrated Train transport pilot - AT), D3.1 (Car Sharing in Smart Cities pilot- S2C), and D4.1 (Information Sharing and Analytics Centre pilot - ISAC). The three pilots represent information *prosumer* (i.e., producer and consumer) interested in sharing data and performing collective analytics using advanced data analysis techniques (WP7) and advanced security services (WP8).

The deliverable starts with an overview of how the E-CORRIDOR framework will work. Then functional requirements are analysed concerning data sharing (i.e., how we regulate the data exchange between parties), data analytics (i.e., how we extract knowledge from the shared data and under which conditions) and data manipulation capabilities (i.e., how we transform data during its sharing and analysis life cycle), as well as non-functional requirements, including security, operational, performance, and usability needs.

D5.1 presents the set of common requirements that will be considered for designing the core architecture of the E-CORRIDOR framework (whose first version will be delivered at M12). Such a reference architecture will then be implemented and made available for pilots' usage and validation (first version at M24).

The deliverable concludes illustrating the requirements for the development, test bed and production environments that will host the E-CORRIDOR reference implementation. It includes both infrastructure requirements (e.g., virtual hardware), development tools (e.g., code versioning and continuous integration service) and security tools to help making the E-CORRIDOR framework more resilient to cyber-attacks.

Table of contents

1. Introduction

This deliverable represents the first steps in designing the E-CORRIDOR framework and plan its subsequent implementation. The framework will be constituted by a set of coordinated services providing the functionalities required for satisfying the goals expressed by the pilots. The pilots will then integrate their own services and components within the E-CORRIDOR framework to include their unique functionalities for data sharing and analysis.

The first step considers eliciting a set of pilot-driven requirements for the framework (objective of this deliverable); next steps will be to design a coherent, generic, confidential and extendible architecture (in two iterations, D5.2 and D5.4); and then provide a reference E-CORRIDOR framework implementation (in two iterations, D5.3 and D5.5).

1.1. Overview of the E-CORRIDOR Framework

The E-CORRIDOR framework will provide a set of services to be used in the multi-modal transport scenarios of the project pilots. The pilots will integrate their own infrastructures, tools and applications within the E-CORRIDOR framework to exploit the framework services.

The actors of the system are referred to as *prosumers*, which are information producers and consumers at the same time. In fact, the basic asset is the information, or data, that the framework allows sharing between a federation of different stakeholders or *parties*.

E-CORRIDOR plans to regulate how this information sharing happens among the different parties. The importance of sharing data between parties becomes very clear when considering that joining data enables more sophisticated analysis and allows obtaining outcomes that are more comprehensive and meaningful. For example, if we consider data sharing for identifying cyber-security attacks, recognising few suspicious data events at a single party could not reveal much information and provide only a limited view. But through observation and correlation of those events in between different parties (e.g., of the same transport vertical sector) it would be possible to reveal an attack pattern or campaign threatening those set of partners (e.g., the train sector). Due to a better overview on the incoming attacks, it would be possible to design and implement more sophisticated defence mechanisms. It is worth mentioning that a current barrier to deliver a similar scenario is that parties are unwilling to share data if they do not have the means to control the exchange and address their privacy concerns: topics that are focal to the E-CORRIDOR framework.

E-CORRIDOR will provide such capabilities thanks to two main subsystems that jointly support the whole architecture (developed in WP6):

- The **Information Sharing Infrastructure (ISI)**: This subsystem regulates the sharing of data between different parties via specific policies called Data Sharing Agreements (DSA).
- The **Information Analytics Infrastructure (IAI)**: This subsystem allows executing analytics services over the shared data obeying to the sharing and analytics DSA policies constraints.

These subsystems of the E-CORRIDOR framework are the ground on which advanced services will be built:

- **Data analytics techniques** (WP7) based on machine and deep learning to perform classification and prediction based on data collaboratively shared, e.g., driver and passenger identification, itinerary planning, privacy preserving analytics, carbon

footprint analytics, and intrusion detection systems. Please refer to WP7 for the full list of data analytics services.

- **Advanced security services** (WP8) for secure access mechanisms, such as continuous policy-based authentication, identity management and behavioural authentication. Please refer to WP8 for the full list of such services.

A key concept in the E-CORRIDOR framework is constituted by the DSA policies. These are rules that regulate all the aspects of the data life cycle, including:

- Who can access the data (e.g., all the parties, a subset of the sharing parties, third parties)?
- Who can use the data, for how long (e.g., until the DSA does not expire) and for what purpose (e.g., analysis purposes)?
- What are the privacy sharing preferences, e.g., what privacy-preserving technique must be applied before data is shared, what data is used in a specific analytics service, or how the analytics result is consumed? These techniques include anonymization and encryption transformations (including Fully Homomorphic Encryption - FHE) of the data and are referred as Data Manipulation Operations (DMOs).

DMOs are the building block on which privacy-preserving analytics and secure access mechanisms will be developed. These DMOs will be defined in WP6.

The E-CORRIDOR framework will be completely distributed and we will use the concept of sticky policies [1] for binding the DSA to the data object in a secured data container. Several instances of ISI distributed at the edge and at the cloud will allow DSA-controlled information sharing, including for example the execution of DMOs that transform the data at the edge before sending it to the cloud. Similarly, distributed instances of IAI will allow some types of analytics at the edge (e.g., on a car) and some other types at the cloud (e.g., those that are more resource and computing intensive like homomorphic computation).

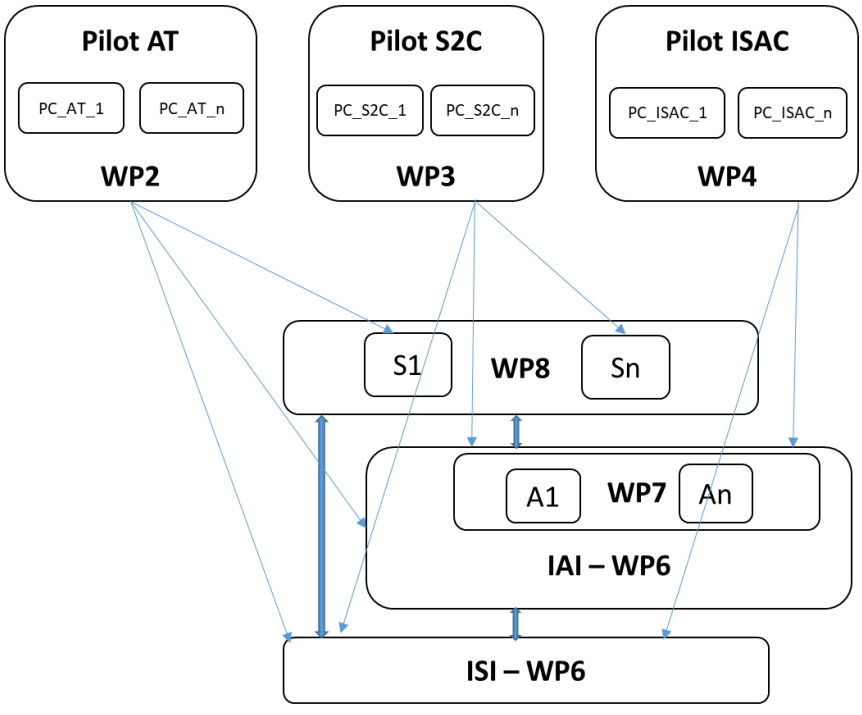

Figure 1: Relationships of the architectural subsystems at macro level

In Figure 1, we describe the main architectural components depicted in WP5, where the E-CORRIDOR reference architecture is developed and the overall framework integrated.

In WP6 we will design the ISI and IAI, considering that the IAI is mainly a subsystem for analytics and the specific analytics functions (A1 … An) are developed in WP7. Also, WP8 will develop extended authentication and authorization security services (S1 … Sn) that will use either IAI or ISI as necessary.

The pilots will build their own software components (PC_x_1 … PC_x_n) that will use components in the WPs 6-8.

1.2. Requirements Naming Convention

We define a naming convention to have a reference to each requirement and make it convenient for tracing it in this document and in future documents. This will allow to easily refer to a requirement and to map each of them to the components of the E-CORRIDOR architecture that will be released at M12. In the next sections, requirements have also a link to the pilot use case or requirement; at a later stage (D5.2), this will allow us to have all the requirements tracing chain from the pilot to the E-CORRIDOR architectural component or module.

Requirements follow the MoSCoW[1] scale [2] for priority classification.

The naming convention is the following:

E-CORRIDOR-[**ReqClass**]-[**Id**]

Where [**ReqClass**] can be:

- A functional class
  - **DS**=Data Sharing, **DA**=Data Analytics, **DM**=Data Manipulation
- A non-functional class
  - **Sec**=Security, further specified as: Sec-**IS** - Information Security, Sec-**RC** - Regulatory/Compliance
  - **Ope**=Operational (distributed computing, extensibility and interoperability)
  - **Per**=Performance
  - **Use**=Usability
  - **Dev**=Development, **Tst**=Test Bed (for E-CORRIDOR system integration activities), **Prd**=Production (E-CORRIDOR computing environments for project implementation for the pilots).

And [**Id**] is a progressive integer number.

1.3. Deliverable Structure

The document is structured as follows:

- Chapter 1 presents an introduction to the E-CORRIDOR framework
- Chapter 2 describes an analysis of the collected requirements from E-CORRIDOR pilots (WP2, WP3, WP4)
- Chapter 3 lists both the functional and non-functional requirements for the E-CORRIDOR framework

[1] "Must have", "Should have", "Could have", and "Won't have but would like"

- Chapter 4 describes the needs for E-CORRIDOR computing environments for development, test and production
- Chapter 5 concludes the document by specifying the next steps of WP5 project activities.

1.4. Definitions and Abbreviations

Term	Meaning
ADS-B	Automatic Dependent Surveillance-Broadcast
AT	Airport and integrated Train transport (WP2 pilot)
BCBP	Bar-Coded Boarding Pass
CAN	Controlled Area Network
CAPEC	Common Attack Pattern Enumeration and Classification
CEF	Common Event Format
CPE	Common Platform Enumerations
CTI	Cyber Threat Information
DMO	Data Manipulation Operations
DSA	Data Sharing Agreement
EML	Electronic Mail
eIDAS	electronic Identification, Authentication and trust Services
EU	European Union
FHE	Fully Homomorphic Encryption
GDPR	General Data Protection Regulation
GPS	Global Positioning System
GPX	GPS Exchange Format
IoT	Internet of Thing
IAI	Information Analytics Infrastructure
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
IIT	*Istituto di Informatica e Telematica* at CNR
ISAC	Information Sharing and Analytics Centre (WP4 pilot)
ISI	Information Sharing Infrastructure
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JPEG	Joint Photographic Experts Group
LAS	LASer format

MoSCoW	“Must have”, “Should have”, “Could have”, “Won’t have but would like”
MMT	Multi-Modal Transport
NMEA	National Marine Electronics Association
NFR	Non-Functional Requirement
ODB	On Board Diagnostics
OWASP	Open Web Application Security Project
RFID	Radio-frequency identification
RSSI	Received Signal Strength
SAML	Security Assertion Markup Language
S2C	Car Sharing in Smart Cities (WP3 pilot)
SSO	Single Sign On
STIX	Structured Threat Information eXpression
UC	Use Case
US	User Story
VM	Virtual Machine

2. Pilots Requirements Analysis

This section summarises the requirements elicitation for the three pilots of E-CORRIDOR - namely, Airport and Train (AT), Car Sharing (S2C) and Information Sharing and Analytics Centre (ISAC). Each pilot considers the application of the E-CORRIDOR concept to a multi-modal transportation ecosystem consisting by air, train and car transportations. All three pilots have special requirements with respect to data sharing and protection, analytic services and passenger experience.

This section groups, at both functional and non-functional level, each of the collected pilots' requirements with the aim of highlighting and grouping by "requirements area" the specifications according to which the E-CORRIDOR framework must be defined. In fact, there are several common E-CORRIDOR needs expressed in the pilots, even if sometimes they use different terms or concepts. To design and build the E-CORRIDOR framework, requirements will need to be extracted from the pilots by using a consistent terminology and common meaning, to synthesise the needs of the different stakeholders. The final goal will be to conceive an E-CORRIDOR framework that is "generic enough" to run all the pilots and possibly new or generalised use cases.

We clustered pilots' requirements in areas by mapping user stories and use cases. These areas where discussed with pilots' owners during teleconferences to come to a common understanding. They are split as follows:

- Functional Requirement:
  - **Data Sharing**: how to collect and share data among parties
  - **Data Analytics**: analysis services executed on collected and shared data
  - **Data Manipulation**: changes to the data content for several purposes (e.g., encryption or anonymization)
- Non-Functional Requirements:
  - **Security**: aspects related to security needs
  - **Operational**: concerning the deployment models for the framework services
  - **Performance**: indications about speed or lack of delay of the expected services
  - **Usability**: includes user experience and friendliness.

These areas will be further detailed in Section 3. In the next section we link use cases (UC) and user stories (US) by following the identifiers used in the pilots' deliverables D2.1 (e.g., AT-UC-01), D3.1 (e.g., S2C-US-01), and D4.1 (e.g., ISAC-UC-01).

2.1. Combined Pilots Requirements

Based on the areas identified above, the combined requirements catalogue formulated from the functional point of view is expressed in Table 1.

Table 1 – Common Functional Pilots Requirements

Area	Use Cases	User Stories
Data Sharing	AT-UC-05	AT-US-01
	AT-UC-09	AT-US-02
	AT-UC-11	AT-US-03
	AT-UC-12	AT-US-04
	AT-UC-14	AT-US-05

	S2C-UC-01 S2C-UC-04 S2C-UC-05 S2C-UC-06 S2C-UC-07 ISAC-UC-01 ISAC-UC-02 ISAC-UC-03 ISAC-UC-06	AT-US-06 AT-US-07 S2C-US-01 S2C-US-04 S2C-US-05 S2C-US-06 S2C-US-07 S2C-US-08 ISAC-US-01 ISAC-US-02 ISAC-US-07 ISAC-US-09
Data Analytics	AT-UC-01 AT-UC-02 AT-UC-03 AT-UC-04 AT-UC-07 AT-UC-08 AT-UC-10 AT-UC-12 AT-UC-13 S2C-UC-02 S2C-UC-03 S2C-UC-06 ISAC-UC-04 ISAC-UC-05 ISAC-UC-07	AT-US-01 AT-US-02 AT-US-03 AT-US-04 AT-US-05 AT-US-06 AT-US-07 S2C-US-02 S2C-US-03 S2C-US-06 ISAC-US-03 ISAC-US-04 ISAC-US-06 ISAC-US-08
Data Manipulation	AT-NFR-02 AT-NFR-05 ISAC-UC-02 ISAC-UC-03	AT-US-01 AT-US-02 AT-US-03 AT-US-04 AT-US-05 AT-US-07 ISAC-US-07 ISAC-US-09

The next table expresses the Non-Functional catalogue of combined requirements:

Table 2 – Common Non-Functional Pilots Requirements

Area	Use Cases	User Stories
Security	AT-NFR-01 AT-NFR-03 AT-NFR-04 AT-NFR-06 AT-NFR-07 AT-NFR-15 AT-NFR-16 AT-UC-06 S2C-NFR-01 S2C-NFR-02 S2C-NFR-03 S2C-NFR-04 S2C-NFR-05 S2C-NFR-06 S2C-NFR-07 ISAC-NFR-01 ISAC-NFR-02 ISAC-NFR-03	AT-US-01 AT-US-02 AT-US-03 AT-US-04 AT-US-05 AT-US-06 AT-US-07 ISAC-US-01 ISAC-US-02 ISAC-US-04 ISAC-US-05 ISAC-US-07 ISAC-US-09
Operational	AT-NFR-08 AT-NFR-09 AT-NFR-13 AT-NFR-14 ISAC-NFR-04 ISAC-NFR-05 ISAC-NFR-06 ISAC-NFR-07	AT-US-02 AT-US-03 AT-US-04 AT-US-05 AT-US-06 AT-US-07 ISAC-US-03 ISAC-US-04 ISAC-US-06 ISAC-US-07 ISAC-US-08 ISAC-US-09
Performance	AT-NFR-12 S2C-NFR-08 S2C-NFR-09	AT-US-01 AT-US-02 AT-US-03

	S2C-NFR-10 ISAC-NFR-08 ISAC-NFR-09 ISAC-NFR-10 ISAC-NFR-11	AT-US-05 AT-US-07 ISAC-US-03 ISAC-US-06 ISAC-US-08
Usability	AT-NFR-10 AT-NFR-11 AT-NFR-12 S2C-NFR-11 S2C-NFR-12 S2C-NFR-13 S2C-NFR-14 ISAC-NFR-12	AT-US-01 AT-US-02 AT-US-03 AT-US-05 AT-US-07 ISAC-US-07 ISAC-US-09

3. Framework Requirements

This section describes the requirements for the E-CORRIDOR framework elicited from the three project pilots and published in their corresponding deliverables (D2.1, D3.1, and D4.1). They are divided in two major classes: functional and non-functional requirements.

3.1. Functional Requirements

We first introduce the concept of **Data Class**, which lists all the different information that will be shared with and used by the E-CORRIDOR provided services.

We then report the functional requirements in the main areas related to the core E-CORRIDOR functionalities that the framework must provide, as resulted from the pilots' requirements analysis and elaboration:

- **Data Sharing**: Relates to the capabilities of sharing data between *prosumers* through the usage of DSAs and associated policies.
- **Data Analytics**: Concerns the possibility to run analytics services that use the shared data and produce a result, where both actions must obey to the applicable DSAs.
- **Data Manipulation Operations**: Describes a set of operations that can be applied to both the shared data and the execution of analytics services as well as their results, in order to transform or manipulate (fields of) the data to address specific needs, including privacy requirements or trustworthiness.

When possible, we mark some requirements as *GDPR requirement*, when the pilot explicitly indicates the need to have a mechanism for complying with some regulatory prescription. Requirements are presented by using the MoSCoW notation for prioritisation, evaluated and adjusted with respect to the pilots' specification.

3.1.1. Data Classes

When sharing data between parties, it is necessary to correctly understand the *resources*, or *data types*, that are the object of the information sharing infrastructure. Starting from analysing each pilots' scenarios, we collected the **list of data types each pilot needs** to share for its business case and at which level of granularity it requires the framework to operate (i.e., a single file, a set of files, or a record in a file).
According to the pilots' use cases, the resources to be shared and protected by E-CORRIDOR framework are of different types. A resource has usually a **data format representation** that describes how it is internally structured with the aim of having both a clear syntax and possibly a clear semantic. The E-CORRIDOR framework should handle different data types, and for the sake of efficiency the resources of the same data type should have the same data format. Using few common data formats enables the framework to be more generic and be used in scenarios that are beyond those conceived by the project pilots. Using a common data format also allows analytics services and data manipulation operations to be reusable in different contexts. Standard structured formats allow easier, efficient and effective representation of resources in the E-CORRIDOR framework.

The following table summarises the data types involved in the specific pilot scenarios (more extended lists are available in D2.1, D3.1, and D4.1). It also shows the use of such data in the three

last columns on the left (for sharing, for analysis, for performing a Data Manipulation Operation). It will be refined during the project.

Table 3 – Data Classes

Data Type Class	Data Format	Standard	Pilot Use Case ID	Sharing	Analysis	DMO
Cyber Threat Information (CTI)	STIX (vulnerabilities, CAPEC, CPE, exploits, attack pattern)	Open Standard	ISAC-US-01 ISAC-US-03 ISAC-US-04 ISAC-UC-01 ISAC-UC-04	✓	✓	
Can bus	CAN Frame	Standard ISO 11898-1	ISAC-US-02 ISAC-US-06 ISAC-US-07 ISAC-UC-02 ISAC-UC-03 ISAC-UC-04 ISAC-UC-07	✓	✓	✓
GPS Data	GPX (GPS Exchange Format)	Open Standard	AT-UC-02 AT-UC-03 AT-UC-04 AT-UC-09 AT-UC-11 AT-UC-12 AT-UC-13 AT-UC-14	✓	✓	
GPS data from smartphone	NMEA 0183/GPRMC sentence <Time, Status, Latitude, Longitude, Speed, Angle, Date, Variation, Integrity, Checksum>	Industry Standard	AT-UC-02 AT-UC-03 AT-UC-04 AT-UC-06 AT-UC-08 AT-UC-13 AT-UC-14	✓	✓	
Boarding pass	BCBP (bar-coded boarding pass)	Industry Standard (IATA)	AT-UC-03 AT-UC-06 AT-UC-07 AT-UC-08	✓		✓
Passport	ICAO9303	Industry Standard	AT-UC-03 AT-UC-06	✓		✓

Image (for facial, fingerprint, or iris recognition) in passport	JPEG, JPEG2000	Open Standard (JPEG2000)	AT-UC-03 AT-UC-06 AT-UC-08 AT-UC-13	✓	✓	✓
Accelerometer	JSON: <time, x, y, z> in m/s^2	Open Standard	AT-UC-03 AT-UC-08 AT-UC-13	✓	✓	✓
Gyroscope	JSON: <time, x, y, z>	Open Standard	AT-UC-03 AT-UC-08 AT-UC-13	✓	✓	✓
Magnetometer	JSON: <time, x, y, z> in uT	Open Standard	AT-UC-03 AT-UC-08 AT-UC-13	✓	✓	✓
Bluetooth RSSI (Received Signal Strength Indication)	JSON: <time, station id, RSSI>	Open Standard	AT-UC-02 AT-UC-03 AT-UC-04 AT-UC-08 AT-UC-12 AT-UC-13 AT-UC-14	✓	✓	
WiFi RSSI (Received Signal Strength Indication)	JSON: <time>, <station id>, <RSSI>	Open Standard	AT-UC-02 AT-UC-03 AT-UC-04 AT-UC-08 AT-UC-12 AT-UC-13 AT-UC-14	✓	✓	
Camera	H.264, .mp4	Open Standard	AT-UC-02 AT-UC-03 AT-UC-04 AT-UC-08 AT-UC-12 AT-UC-13 AT-UC-14	✓	✓	✓
Lidar	LAS	Open Standard	AT-UC-02 AT-UC-03 AT-UC-04 AT-UC-08	✓	✓	✓

			AT-UC-12 AT-UC-13 AT-UC-14			
RFID	Raw bits	No Standard	AT-UC-06	✓		✓
Passenger data	JSON: <name, surname, date of birth, place of birth, nationality>	Open Standard	AT-UC-06 AT-UC-14	✓		✓
Network log	syslog-ng[2]	Open Standard	AT-UC-09 AT-UC-10 ISAC-US-02 ISAC-US-03 ISAC-US-08 ISAC-US-09 ISAC-UC-02 ISAC-UC-03 ISAC-UC-04 ISAC-UC-06 ISAC-UC-07	✓	✓	✓
Event log	CEF[3]	Open Standard	AT-UC-09 AT-UC-10 ISAC-US-02 ISAC-US-03 ISAC-US-08 ISAC-US-09 ISAC-UC-02 ISAC-UC-03 ISAC-UC-04 ISAC-UC-06 ISAC-UC-07	✓	✓	✓
Network log	NetFlow[4]	Industry Standard (CISCO)	AT-UC-09 AT-UC-10 ISAC-US-02	✓	✓	✓

[2] https://www.syslog-ng.com/

[3] https://community.microfocus.com/dcvta86296/attachments/dcvta86296/connector-documentation/1197/2/CommonEventFormatV25.pdf

[4] https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html

			ISAC-US-03 ISAC-US-08 ISAC-US-09 ISAC-UC-02 ISAC-UC-03 ISAC-UC-04 ISAC-UC-06 ISAC-UC-07			
Airplane tracking	ADS–B (Automatic dependent surveillance-broadcast)	Industry Standard	AT-UC-09 AT-UC-10 AT-UC-11	✓	✓	✓
On Board Diagnostics	OBD	Industry Standard?	ISAC-US-06 ISAC-US-07 ISAC-UC-02 ISAC-UC-03 ISAC-UC-04 ISAC-UC-07	✓	✓	✓
Bus geo-positioning	GPX	Proprietary	S2C-US-06	✓		
Bus speed	Proprietary	Proprietary	S2C-US-06	✓		
Time of arrival of the bus at every stop	Proprietary	Proprietary	S2C-US-06	✓		
User profile	JSON	Proprietary	S2C-US-01 S2C-US-06 S2C-US-06 S2C-US-07 S2C-US-08	✓	✓	
Trip data	JSON	Proprietary	S2C-US-02 S2C-US-06 S2C-US-03	✓	✓	✓
Micro-subsidies calculation results data	JSON	Proprietary	S2C-US-02	✓	✓	
Connection logs	Structured Text	Proprietary	S2C-US-06 S2C-US-07		✓	✓
E-mails	EML	Open Standard	S2C-US-06 S2C-US-07		✓	✓

DRT (bus-on-demand) service and usage data	JSON	Proprietary	S2C-US-06	✓		

3.1.2. Data Sharing Requirements

The next table consolidates the pilots' needs in terms of capabilities required for the **Information Sharing Infrastructure** (ISI) of E-CORRIDOR, including necessities of security policies to regulate data access and usage, both at the edge and at the cloud.

Table 4 – Data Sharing Requirements

ID	Goal	Priority	Requirement
E-CORRIDOR-DS-01	AT-NFR-13	MUST	The E-CORRIDOR framework provides a way to define **Data Sharing Agreements** between parties.
E-CORRIDOR-DS-02	AT-US-04	MUST	E-CORRIDOR allows defining **multi-lateral DSAs**, i.e., between multiple parties (more than two).
E-CORRIDOR-DS-03	ISAC-US-05 ISAC-US-09	MUST	E-CORRIDOR allows defining **policies** as a set of rules that regulate the data **sharing process** expressed in the DSA.
E-CORRIDOR-DS-04	ISAC-US-07 ISAC-US-09	MUST	E-CORRIDOR policies in the DSAs allow specifying conditions for **authorizations** (CAN), **obligations** (MUST), and **prohibition** (MUST NOT) logics.
E-CORRIDOR-DS-05	AT-US-03	MUST	E-CORRIDOR allows sharing data (the "object" protected by the DSA) of **different formats**.
E-CORRIDOR-DS-06	AT-US-01	MUST	E-CORRIDOR policies consider **end-user properties** within their logic.
E-CORRIDOR-DS-07	AT-US-01	MUST	E-CORRIDOR policies consider **context**/**environment** properties within their logic.
E-CORRIDOR-DS-08	S2C-US-04	MUST	E-CORRIDOR policies consider **time-based conditions** within their logic.
E-CORRIDOR-DS-09	AT-US-02	MUST	E-CORRIDOR policies enable to express conditions to preserve **confidentiality** over the shared data.
E-CORRIDOR-DS-10	AT-US-02	MUST	E-CORRIDOR policies enable to express **retention criteria** over the shared data (e.g., shared data is deleted after a certain amount of time or at a fixed date). (GDPR Requirement)

E-CORRIDOR-DS-11	AT-US-02	MUST	E-CORRIDOR policies enable to express conditions to control **access** to the shared data (i.e., conditions to evaluate before obtaining the data).
E-CORRIDOR-DS-12	AT-US-02	MUST	E-CORRIDOR policies enable to express conditions to control **usage** of the shared data (i.e., continuous authorisation after granted access).
E-CORRIDOR-DS-13	AT-US-02	MUST	E-CORRIDOR allows writing DSA policies for activating "**pre-processing rules**" on shared data, which are data manipulation operations (DMOs) performed before data is shared with other prosumers.
E-CORRIDOR-DS-14	AT-US-02	MUST	E-CORRIDOR data object preserves **ownership** of its stakeholder (i.e., the object data creator or the data collecting entity).
E-CORRIDOR-DS-15	AT-US-02	MUST	E-CORRIDOR provides **DSA templates** to be used as pre-established or ad-hoc agreements to be instantiated by parties into DSAs.
E-CORRIDOR-DS-16	AT-UC-05	MUST	E-CORRIDOR provides **dynamic data sharing user preferences** to be configured by the producer (end-user), in an opt-in/opt-out fashion at data creation time.
E-CORRIDOR-DS-17	AT-UC-09 S2C-US-07	MUST	E-CORRIDOR allows data producers to **accept or reject the DSA**, enabling to show the **parties** that will have access to the data, for which **purpose** and how the consumer(s) can **operate on** the data. (GDPR Requirement)
E-CORRIDOR-DS-18	S2C-US-08	MUST	E-CORRIDOR enables **showing** to the producer **the DSA** applied to the data.
E-CORRIDOR-DS-19	ISAC-US-01	MUST	E-CORRIDOR receives data (**push mechanism**) from external sources (in particular CTI).
E-CORRIDOR-DS-20	ISAC-US-01	MUST	E-CORRIDOR **pulls data** (in particular CTI) from external sources, with specified polling intervals.
E-CORRIDOR-DS-21	ISAC-US-02	MUST	E-CORRIDOR allows **using services** (i.e., access to them by API) of the Information Sharing Infrastructure **based on policy constraints**.

E-CORRIDOR-DS-22	ISAC-US-04 ISAC-UC-05 S2C-US-08	SHOULD	E-CORRIDOR allows defining **notifications policies** in the DSA that send a notification event (e.g., e-mail to a specified user or SMS) when the analytics service **result** is **generated** or have a **specific value** (e.g., IDS found a malicious data).
E-CORRIDOR-DS-23	S2C-US-08	MUST	E-CORRIDOR allows defining **notification policies** at data **ingestion** time (create/upload) and at data **read** time.
E-CORRIDOR-DS-24	S2C-US-08	SHOULD	E-CORRIDOR allows defining **notification policies** that use the **result data or metadata** as notification body text.
E-CORRIDOR-DS-25	ISAC-US-04	MUST	E-CORRIDOR attaches the DSA policies to the **shared data** (**sticky policy approach**).
E-CORRIDOR-DS-26	ISAC-UC-08 S2C-US-01 S2C-US-07	MUST	E-CORRIDOR allows **searching the collected data** by meta data set on the shared data (e.g., by data owner, party/parties collecting the data, purpose of sharing, date of sharing, and type/class of data).
E-CORRIDOR-DS-27	AT-UC-14	MUST	E-CORRIDOR allows **searching the DSAs** that are available for use depending on specific DSA properties (e.g., by parties in the agreement).
E-CORRIDOR-DS-28	S2C-US-01	SHOULD	E-CORRIDOR allows **updating shared data** by an authorised party (e.g., producer updating the data content).
E-CORRIDOR-DS-29	S2C-US-01	MUST	E-CORRIDOR keeps a **logbook** of when the **operations on data** took place (e.g., create or read).
E-CORRIDOR-DS-30	S2C-US-01	SHOULD	E-CORRIDOR allows **notifying parties** of the DSA that a data object has been updated (via a log message). Update is considered as a delete followed by a create operation.
E-CORRIDOR-DS-31	S2C-US-01	MUST	E-CORRIDOR allows **revoking the DSA** to deny the access to all the related data.

3.1.3. Data Analytics Requirements

The E-CORRIDOR framework will provide data analytics as a service to support the pilots analysis needs, both at the cloud and at the edge. In fact, a data analytics is a **computational operation** that can be executed over the shared data between a set of prosumers under the regulation of DSA policies. These requirements provide input to the definition of the **Information Analytics Infrastructure** (IAI).

Analytics requirements come from joint analysis work between WP7 and pilots WPs (WP2/3/4) where WP7 inspected the pilots' needs/use cases and figured out the benefits they could bring to pilots, while pilots suggested needs for investigation and data analysis.

Table 5 – Data Analytics Requirements

ID	Goal	Priority	Requirement
E-CORRIDOR-DA-01	ISAC-US-02	MUST	E-CORRIDOR allows defining **policies**, as a set of rules that regulate the data **analytics execution** over the shared data (e.g., on who can run a specific analytic and under which conditions).
E-CORRIDOR-DA-02	AT-US-03 AT-UC-10 ISAC-US-02 ISAC-UC-06 ISAC-NFR-05	MUST	E-CORRIDOR provides a way to transform data into a **common data format** before collecting it (in this way analytics services will work on the same format regardless of the data source). The format is related to the class or type of data (see Table 3 – Data Classes).
E-CORRIDOR-DA-03	AT-US-04 AT-US-06 ISAC-US-05 ISAC-US-07 ISAC-US-09	MUST	E-CORRIDOR policies enable to express data sharing conditions over the **analytics results**.
E-CORRIDOR-DA-04	AT-US-04	MUST	E-CORRIDOR policies over the analytics results include conditions on the **stakeholder relevance** (e.g., distribute results to stakeholders according to access control rules), by means of access control rules defined in the DSA policies.
E-CORRIDOR-DA-05	AT-US-04 AT-US-06	MUST	E-CORRIDOR allows running analytics **over data shared** by different parties, **according to (i.e., satisfying) individual parties' policies** in the DSA.
E-CORRIDOR-DA-06	AT-US-04	MUST	E-CORRIDOR allows **sharing analytics results** to parties in the DSA (e.g. it allows reading an analytics result/outcome).
E-CORRIDOR-DA-07	AT-US-06	MUST	E-CORRIDOR allows writing DSA policies for activating "**post-processing**

			rules" on an analytics operation result, which are data manipulation operations (DMOs) performed before delivering the result to the prosumer.
E-CORRIDOR-DA-08	ISAC-US-04	MUST	E-CORRIDOR attaches the DSA policies to the **analytics result** data (**sticky policy approach**).
E-CORRIDOR-DA-09	ISAC-UC-08	MUST	E-CORRIDOR allows **searching the generated analytics results data** (e.g., by owner/party, time, etc.).
E-CORRIDOR-DA-10	S2C-US-09	MUST	E-CORRIDOR provides **encrypted communication channels** to clients.
E-CORRIDOR-DA-11	S2C-US-09	SHOULD	E-CORRIDOR allows using Fully Homomorphic Encryption (**FHE**) based analytics.

3.1.4. Data Manipulation Operations

In addition to analytics operations, the E-CORRIDOR framework allows one to define data operations that modify the shared data under specific constrains (expressed through the DSA policies). **Data Manipulation Operations** (DMOs) are used to *pre-process* the information before or *post-process* after usage and analytics to make it usable for further processing, or to comply with the associated sharing policy. For example, we can specify a DSA policy with an obligation to perform a listed DMO when the data is read by "Party A" or by "users with role *analyst*". DMOs are typically used to implement Privacy-Preserving Techniques to satisfy data sanitization requirements over the shared data or the analytics service result. Privacy-Preserving Techniques apply usually to protect personal data, as in the context of measured suggested for regulatory compliance (e.g., EU GDPR 2016/697 regulation). They include **anonymization** (techniques allowing to scrub any information useful for data subject identification), **pseudo-anonymization** (also referred to as *pseudonymization*, techniques with the peculiarity to de-identify personal information with pseudonyms still allowing an off-line indirect re-identification), and **encryption** (scrambled text than can be put again in clear by using a key maintained secret to some people or systems; this includes *homomorphic encryption*).

Table 6 – Data Manipulation Requirements

ID	Goal	Priority	Requirement
E-CORRIDOR-DM-01	AT-UC-09 ISAC-US-02 S2C-US-05 S2C-US-10	MUST	E-CORRIDOR allows **data obfuscation** or **anonymization** of specific data fields (e.g., for biometrical user data). (GDPR Requirement)
E-CORRIDOR-DM-02	AT-UC-09 ISAC-US-02 ISAC-UC-03 ISAC-NFR-03	MUST	E-CORRIDOR allows **data pseudo-anonymization** of specific data fields. (GDPR Requirement)

E-CORRIDOR-DM-03	AT-NFR-02 ISAC-UC-03	MUST	E-CORRIDOR allows **data encryption** of specific data fields (e.g., for biometrical user data).
E-CORRIDOR-DM-04	S2C-US-09	SHOULD	E-CORRIDOR allows Fully Homomorphic Encryption (**FHE**) of data (email addresses or IP addresses).

3.2. Non-Functional Requirements

Non-Functional Requirements (NFRs) are important for building a framework that not only works but operates by using high-quality standards and best practices. The requirements are grouped in the following sets:

- **Security**: Since E-CORRIDOR focuses strongly on security aspects, Information Technology security requirements, including privacy and the regulatory needs are of prime importance
- **Operational**: This set of requirements define specific characteristics of the environments where the pilots operate
- **Performance**: Pilots have requirements related to the impact on their operative flows, e.g., concerning the maximum amount of time users will wait for specific services. E-CORRIDOR analytics services and DMOs (including homomorphic computing and encryption) might have high memory and computation necessities
- **Usability**: The E-CORRIDOR framework should provide services that are effective and easy to consume, as well as it should enable pilots to create such kind of services.

3.2.1. Security Requirements

This section reports on E-CORRIDOR requirements that address properties of **confidentiality**, **integrity** and **availability** (CIA), as well as **authentication**, **authorisation**, **non-repudiation**, and **accountability**.

E-CORRIDOR framework needs to assure the following properties:

- **Confidentiality** (i.e. secrecy) of shared data by prosumers, as well as of analytics results. It further should enable to apply different degrees of confidentiality measures depending on the specific scenario
- **Integrity** of the shared data and analytics results, assuring data is modified only when it needs to be (e.g., by policy-controlled data manipulation operations)
- **Availability** of data in order not to lose shared data or analytics results
- **Authentication** and **authorisation** to allow guaranteeing that access to data and analytics results is permitted only once an identity is securely established and it is evaluated to have the proper rights to access the data
- **Non-repudiation** and **accountability** to help litigation resolution and to avoid misconducts (e.g., a party denies to sharing some data or use some analytics result). It also allows meeting compliance to legislative requirements or other compulsory regulations.

The following tables lists the **IT Security requirements**.

Table 7 – Security Requirements for Information Security

ID	Goal	Priority	Requirement
E-CORRIDOR-Sec-IS-01	AT-US-02 AT-NFR-01	MUST	E-CORRIDOR traces with **audit trails** the DSA policies evaluations, like authorisations outcomes (e.g. grant or deny access to a shared data), analytics execution, and data manipulation operations, for auditing purposes including accountability, non-repudiation and compliance.
E-CORRIDOR-Sec-IS-02	AT-NFR-05 ISAC-NFR-03	MUST	E-CORRIDOR stores **data encrypted at-rest** to preserve confidentiality and privacy.
E-CORRIDOR-Sec-IS-03	AT-NFR-06 ISAC-NFR-01 S2C-US-09	MUST	E-CORRIDOR **protects data in-transit** (e.g., using TLS protocol) with **encrypted channels** to collect (e.g., upload) or deliver data (e.g., read), allowing to preserve confidentiality, privacy and authenticity.
E-CORRIDOR-Sec-IS-04	AT-NFR-06 ISAC-NFR-02	MUST	E-CORRIDOR employs **data integrity** measure over the shared data.
E-CORRIDOR-Sec-IS-05	AT-NFR-06	SHOULD	E-CORRIDOR uses capabilities to evaluate the **integrity** of the running framework.
E-CORRIDOR-Sec-IS-06	AT-NFR-16	MUST	E-CORRIDOR provides its **API** functionalities after performing **authentication** and **authorisation** steps by using standard protocols (e.g., OpenID Connect, OAuth2).

The next table contains the **Regulatory/Compliance requirements**.

Table 8 – Security Requirements for Regulatory/Compliance

ID	Goal	Priority	Requirement
E-CORRIDOR-Sec-RC-01	AT-US-02	MUST	E-CORRIDOR enables to define policies for **being compliant** with (some[5]) prescriptions of regulations (e.g. GDPR) and privacy needs. For example, these policies could allow expressing functional needs of access control or data

[5] We understand that being compliant with a regulation is not only a matter of DSA policies, but here we focus on policies that can support some regulatory prescriptions (e.g., data pseudo-anonymization or data retention).

			anonymization, as those presented in Sections 3.1.2, 3.1.3, 3.1.4.

3.2.2. Operational Requirements

E-CORRIDOR framework is completely distributed and shall work both at the edge and at the cloud for data sharing and analysis. We split between distributed computing, and extensibility and interoperability requirements.

Table 9 – Distributed Computing Requirements

ID	Goal	Priority	Requirement
E-CORRIDOR Ope-01	AT-US-06 ISAC-US-06	MUST	E-CORRIDOR provides a distributed computing deployment model both at the **edge** and at the **cloud**.
E-CORRIDOR Ope-02	AT-NFR-14 ISAC-US-06 ISAC-US-08 ISAC-UC-07	MUST	E-CORRIDOR provides **analytics at the edge** capability.
E-CORRIDOR Ope-03	ISAC-US-06 ISAC-US-08 ISAC-UC-07	MUST	E-CORRIDOR provides **collaborative analytics** at the **cloud**.

In the next table we report requirements to enable the integration with the E-CORRIDOR framework to build a system that is both easily extensible and interoperable.

Table 10 – Extensibility and Interoperability Requirements

ID	Goal	Priority	Requirement
E-CORRIDOR Ope-04	ISAC-UC-04	MUST	E-CORRIDOR provides a **standard way** (e.g., guidelines, API, code skeleton template, etc.) for creating E-CORRIDOR **compliant analytics services**.
E-CORRIDOR Ope-05	ISAC-NFR-04	SHOULD	E-CORRIDOR provides an **asynchronous** way to run analytics services.
E-CORRIDOR Ope-06	AT-UC-14	SHOULD	E-CORRIDOR provides a **standard way** (e.g., guidelines, API, code skeleton template, etc.) for **creating a DMO** (Data Manipulation Operations).
E-CORRIDOR Ope-07	AT-UC-14	MUST	E-CORRIDOR provides its functionalities through an Application Programming Interface (**API**) based on the **REST** principle and open standards (e.g., OpenAPI [11]).

E-CORRIDOR Ope-08	E-CORRIDOR-Tst-Int-AT-03 E-CORRIDOR-Tst-Int-AT-04	MUST	E-CORRIDOR framework is delivered via the **micro-services** architectural pattern (based on **containers**).

3.2.3. Performance Requirements

The need to have a system that behaves efficiently yet can provide the expected functionalities can be most of the times a big challenge. In particular, the architecture design should consider from the early beginning such challenge with the main intent to minimise the disruption on the pilots' end-users.

Table 11 – Performance Requirements

ID	Goal	Priority	Requirement
E-CORRIDOR Per-01	AT-US-06	MUST	E-CORRIDOR reduces the **data transfer** between the edge and the cloud.
E-CORRIDOR Per-02	AT-NFR-12	MUST	E-CORRIDOR **authentication mechanism** should have a low performance impact.
E-CORRIDOR Per-03	ISAC-NFR-06	SHOULD	E-CORRIDOR allows sharing (uploading) a **large amount of data**.

3.2.4. Usability Requirements

Usability deals primarily with effectiveness and easiness of use of a solution as well as to be learned to become profitable in its execution with the least amount of time as possible. Usability should be considered both in the presentation aspects of E-CORRIDOR tools (e.g., GUI) and in the simplicity of processes that use such tools.

Table 12 – Usability Requirements

ID	Goal	Priority	Requirement
E-CORRIDOR Use-01	AT-NFR-16	MUST	E-CORRIDOR uses **standard authentication protocols** (e.g. OpenID Connect [7], OAuth2 [8], SAML [9], eIDAS [10])
E-CORRIDOR Use-02	AT-US-02	SHOULD	E-CORRIDOR usage allow seamless authentication by leveraging **Single-Sign On** (SSO) authentication schemata.

4. Requirements for Development, Test Bed and Production Environments

In this section we describe the requirements to achieve the goal of having an E-CORRIDOR reference architecture implementation to be used by pilots for their evaluation.

To address this goal, we foresee to have three computing environments:

- **Development**: This environment is a collection of tools and services that the E-CORRIDOR consortium partners can use to craft the components and modules of the whole framework, as well as to build the pilots artefacts that are needed to bridge, or integrate, the framework with the pilots use cases. It includes source code versioning systems, continuous integration and deployment tools, compilation toolchains, etc.
- **Test Bed**: The Test Bed is where the integration efforts take place and represent the environment where the latest version of the software artefacts are deployed and integrated. By its own nature, it could be **unstable**, but provides always the latest available features
- **Production**: The Production is where a component is deployed when it has been promoted from Test Bed after reaching a good-enough maturity level where both unit and integration testing activities have confirmed its stability. It is a **stable** environment, but could lack the latest framework features, and it is aimed at providing the E-CORRIROR reference architecture implementation for integrating and exercising the pilots use cases, e.g., for demos at conferences, events and final validation.

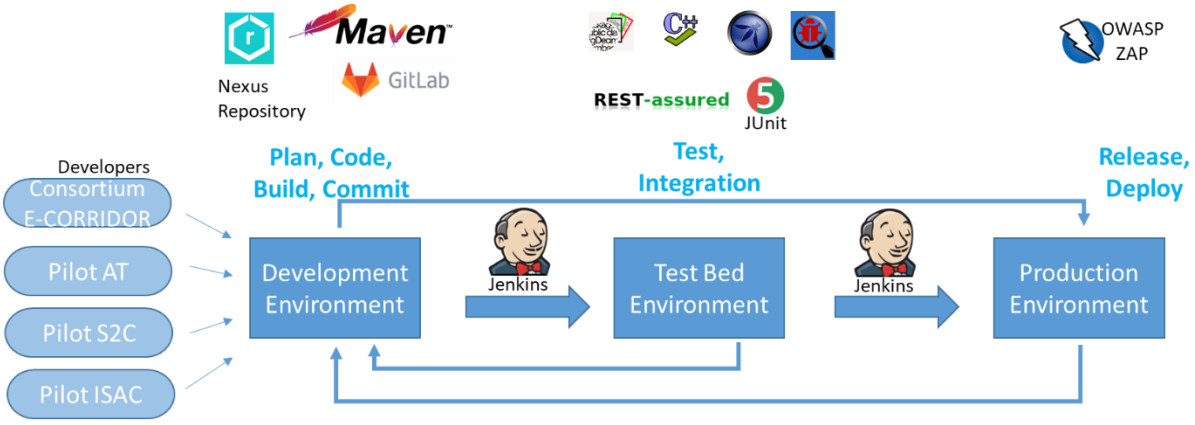

Figure 2: E-CORRIDOR environments

In the next sections we will list the requirements for these environments, both in term of resources and tools.

4.1. Development Environment Requirements

4.1.1. Development Environment Infrastructure Configuration

The Development Environment will be hosted by one Virtual Machine that will run in IIT CNR Data Center. This Data Centre hosts several physical servers running VMware vSphere infrastructure [12], which will host the Virtual Machines of the E-CORRIDOR project.

E-CORRIDOR – Development Environment Configuration		
Number of Machines	**Aim**	**Features**
1 Virtual Machine	This machine will host all the development tools (described in Section 4.1.2) that will be used for the development of the E-CORRIDOR framework	4 CPU Cores 12 GB RAM 200 GB Storage UBUNTU server 20.04 LTS OS

4.1.2. Development Tools

In this section, we describe tools that we intend to deploy to the development environment with the aim to support coding of partners for the reference implementation of the E-CORRIDOR framework.

Tools reported here are as actual state, so, in case of changes due to different requirements, they will be substituted if necessary or no more fitting.

The tools are selected to supply infrastructure with a continuous integration service in these classes of solutions:

- **Collaborative Code & Version Control System**: This system supports storing the source code of the project along with all the required artefacts (libraries, configuration files, test code, etc.). It provides the following major functionalities: collaborative development by multiple parties (developers), source code merging and code conflicts resolutions (when two or more people update the same section of source code)
- **Build Automation System**: This system help automating the process of building or compiling the source code and packaging it as an executable or binary object. Some major functionalities include the setup of project code structure that follows best practices, and the capability of handling dependencies of software libraries (i.e., find and retrieve the correct version of dependent artefacts)
- **Artefact Repository System**: While the Build Automation System promotes code libraries dependency management, the Artefact Repository is the actual system that keeps the available binary software components and is used to store new artefacts that can be shared among different software projects
- **Testing System**: This system supports both the automation of *unit testing* (testing of methods, classes, modules, etc.) and the *integration testing* (i.e., testing between modules and components) by defining "test cases" that can be easily run at every code build. Test frameworks allow developers to mock-up services (creating code stubs) to

verify the component in isolation, to substitute modules that are not yet available, or to test a complete chain of components end-to-end. Containerisation platforms also support the testing system because they can create ad-hoc or on-demand disposable testing environments

- **Bug Tracking System**: This tool supports people with processes to trace software defects or improvements that are found for the systems components in development and/or related to the deployment environments. It allows setting priorities on found issues and assigning them to specific developers, tracking their status and evolution over time, by typically providing a centralised dashboard
- **Continuous Integration System**: This system orchestrates the build process and coherently executes all the steps performed by the previously described systems. It is the basis of a modern and agile software development life cycle and allows automating the source code retrieval from the Version Control System, its compilation via the Build Automation toolchains, the storage of the generated software components in the Artefact Repository, up to the execution of the testing toolbox, including functional and security tests. This system also integrates with the software containerisation solutions to both packaging the container and deploy it on the container platform. It also assists in the promotion of the software matching the required quality criteria to production by automating the deployment procedures.

The following table summarises the Software Development Tools selected, at this stage of the project, for each of previous class:

E-CORRIDOR - Software Development Tools		
Category	**Tool**	**Tool Website**
Collaborative Code & Version Control	GitLab	https://about.gitlab.com
Build Automation	Apache Maven	https://maven.apache.org
Artefact Repository	Nexus Repository OSS	https://www.sonatype.com/nexus/repository-oss
Testing	JUnit (Unit) RestAssured (Integration)	https://junit.org/junit5 https://rest-assured.io
Bug Tracking	GitLab	https://about.gitlab.com
Continuous Integration	Jenkins	https://www.jenkins.io

The tool selection was based on experience of the consortium partners, tools used in other EU funded projects or internal R&D factories, the fact of being open source, and on public reviews and ratings (e.g., Synopsys Black Duck Open Hub[6] site). Below we summary the major features and capabilities of the selected toolset.

6 https://www.openhub.net

Collaborative Code & Version Control: GitLab

GitLab is an open source software to collaborate on code. GitLab offers git version control repository management, code reviews, issue tracking, activity feeds and wikis. Enterprises install GitLab on-premises and connect it with LDAP and Active Directory servers for secure authentication and authorization of developers and partners (a setup we will use in E-CORRIDOR). A single GitLab server can handle thousands of users but it is also possible to create a high availability setup with multiple active servers.

It has a well-established, mature codebase maintained by a very large development team and few vulnerabilities reported in years from OpenHub. It uses an open source MIT License that is commercially friendly.

Build Automation: Apache Maven

Maven is a software project management and comprehension tool made by the Apache Software foundation under a permissive open source Apache License 2.0. Based on the concept of a project object model (POM), Maven can manage a projects' build, reporting and documentation from a central piece of information. It also supports multi-projects builds, i.e., the capability of building a software project that has been split into many smaller sub-projects.

Maven provides a standard way to build the projects, a clear definition of what the project consists of, an easy way to publish project information, and a way to share artefacts across several projects. The tool can be used for building and managing any Java-based project.

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, Maven deals with several areas of concern:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Encouraging better development practices

Artefact Repository: Nexus Repository OSS

Nexus OSS is an open source repository that supports many artefact formats, including Docker, Java, and NodeJS (npm). With the Nexus tool integration, pipelines in toolchain can publish and retrieve versioned components and their dependencies by using central repositories that are accessible from other environments.

Nexus OSS has a broad support for many tools:

- Store and distribute Maven/Java, npm, Docker, and more
- Manage components from the continuous integration pipeline, such as binaries, containers, assemblies, etc.
- Support the Java ecosystem, including Maven, Gradle, Ant, and Ivy
- Compatible with popular tools like Eclipse, IntelliJ, Jenkins, Docker, and more

It is release under the open source Eclipse Public License 1.0.

Testing (Unit): Junit

JUnit is a mature unit testing framework for the Java programming language. JUnit is an open-source (Eclipse Public License 1.0) framework used to write and run repeatable automated tests. It integrates with Jenkins for reporting the testing outcomes and with Maven for executing the tests.

Testing (Integration): Rest Assured

REST Assured is a Java toolset that provides a domain-specific language (DSL) for writing powerful, maintainable tests for RESTful APIs. Since E-CORRIDOR will be a micro-services-based framework orchestrated in containers, RESTful APIs is the way communication and interaction between micro-services happen. A toolset for testing end-to-end the framework functionalities is of prime importance.

REST Assured is released via the permissive Apache License 2.0.

Continuous Integration: Jenkins

Jenkins is a continuous integration server, allowing to automatically monitor source repositories, build software, run tests and deploying software. Through the installation of plugins, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain, including the GitLab and Maven. It has several dashboards for controlling the status of the unit and integration tests (e.g., JUnit compatible) and dashboards for visualising the status of the quality and security tests performed on code artefacts.

Jenkins is made available via an open source MIT License.

4.1.3. Quality and Assurance Strategy

In every **Software Development Life Cycle** the maturation of a software solution cannot happen without continuously improving its overall quality and security posture. There are several international standards, best practices and maturity models that foster the adoption of system and software quality practices, both general purposes, like ISO/IEC 25010[7] - Systems and software engineering [3], CMMI[8] - Capability Maturity Model Integration [4], or more security oriented like OWASP Software Assurance Maturity Model [6] - SAMM[9], or Building Security In Maturity Model [6] - BSIMM[10].

We selected several tools that will be useful in assessing the quality of the software that is being created for the E-CORRIDOR framework, and in particular tools for performing both **static** and **dynamic code analyses**. The selected tools are all open source and focus on the technologies that at this stage partners are known to be used by the E-CORRIDOR partners, like Java, NodeJS (Javascript) and C/C++ programming languages. The tools' list could be extended during the project lifetime, if necessary.

Not only such tools will be used for the E-CORRIDOR framework, but also artefacts developed by the pilots for their integration, as well as WP7 and WP8 objects, could benefit from their application and adoption. In fact, these tools will be integrated with a **SecDevOps** approach in

7 https://www.iso.org/standard/35733.html

8 https://cmmiinstitute.com/cmmi/intro

9 https://owaspsamm.org

10 https://www.bsimm.com

the Continuous Integration pipeline such in a way that each and every build can be assessed and its security and quality posture determined and used as a factor to enable or not its promotion to integration test bed or production environments.

As with the development tools seen in Section 4.1.2, these tools have been selected considering the experience of the partners in previous software projects, and public reviews or ratings (including the already cited Synopsys Black Duck Open Hub service). In the next paragraphs, we show the selected toolset and illustrate their major capabilities for both static and dynamic code assessment.

1.1.1.1. Static Analysis Tools

In this section we describe tools that will be used for **static software analysis** of E-CORRIDOR software code created on the Development Environment. Static analysis is the practice of analysing the source code of a system without having it running. Such tools are dedicated to Java, C and C++ programming languages.

E-CORRIDOR - Static Analysis Tools		
Category	**Tool**	**Tool Website**
Syntactical checking Java code	CheckStyle	https://checkstyle.sourceforge.io
Static analysis in C/C++ code	Cppcheck	http://cppcheck.sourceforge.net
Dependency Check	OWASP	https://owasp.org/www-project-dependency-check
Static analysis of Java code	SpotBugs	https://spotbugs.github.io

Static Analysis: Checkstyle

Checkstyle is a quality assessments tool to help programmers write Java code that adheres to a coding standard about layout and formatting. It automates the process of checking the compliance of the written source code and so helps enforcing a coding standard. Checkstyle is highly configurable and ships with already defined coding standards such as those from Google or Oracle, and customizable rules that can be configured to support other best practices. Checkstyle can check many aspects of source code and can inspect both at the class and method level. According to Open Hub, CheckStyle is a mature project maintained by a very large community of developers. It has a high Security Confidence Index and a High Vulnerability Exposure Index, which means few vulnerabilities has been reported over time. It integrates with Jenkins and uses the open source GNU Lesser General Public License 2.1.

Static Analysis: Cppcheck

Cppcheck is a free and open source static analysis tool for C/C++ code. It provides code analysis to detect bugs and focuses on detecting undefined behaviour and dangerous coding constructs with the goal of having very few false positives. Cppcheck is able to detect defects such as dead pointers, division by zero, integer overflows, memory management issues, and null pointer dereferences, amongst the others.

Cppcheck can be integrated in a Jenkins job with a plug-in[11] that scans for Cppcheck report files in the build workspace and reports the issues detected during static C/C++ code analysis. It is release under the open source GNU General Public License version 3.0.

Static Analysis: OWASP Dependency-Check

OWASP Dependency-Check[12] is a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a source code project's dependencies. The tool extracts the Common Platform Enumeration (CPE) identifier (if available) of all code libraries (dependencies) used by the source code and it then generates a report pointing the associated Common Vulnerabilities and Exposures (CVE) records. Dependency-Check integrates with the Jenkins continuous integration server and can inspect and report such warnings at every code build. It is based on the OWASP risk #9 Using Components with Known Vulnerabilities, part of the OWASP Top Ten project. It is released with the Apache Software License 2.0 as open source code.

Static Analysis: SpotBugs

SpotBugs[13] uses static analysis to look for bugs in Java code. It is free software, distributed under the terms of the GNU Lesser General Public License[14]. SpotBugs is the successor of FindBugs[15], a tool developed at the University of Maryland, USA since 2007, but now lo longer maintained. SpotBugs carries on from the point where FindBugs left off with support of its community. SpotBugs checks for more than 400 bug patterns[16]. SpotBugs can be used standalone and through several integrations, including Maven[17] and during the continuous integration builds from Jenkins. SpotBugs is extensible through plugins with new *detectors* to add support for spotting new code issues. With respect to security static code analysis a SpotBugs excellent plugin is **Find Security Bugs**[18]. Find Security Bugs detects over 138 different types of security vulnerabilities, which includes and extensive support for the already mentioned OWASP Top Ten security risks. This plugin is release with the same open source license as SpotBugs.

1.1.1.2. Dynamic Analysis Tools

In this section we describe tools that will be used for **dynamic software analysis** of E-CORRIDOR software code created on the Development Environment. Dynamic analysis is the practice of evaluating (security) defects with the system fully running. It is for the most part agnostic with respect to the underling technologies (e.g., programming language), because it tests the running web application or services. Of course, testing (even truer when trying to exploit) some vulnerabilities require to specify constructs using the used technologies (e.g., SQL queries).

---

11 https://plugins.jenkins.io/cppcheck/

12 https://owasp.org/www-project-dependency-check/

13 https://spotbugs.github.io/

14 http://www.gnu.org/licenses/lgpl.html

15 http://findbugs.sourceforge.net/

16 https://spotbugs.readthedocs.io/en/latest/bugDescriptions.html

17 http://spotbugs.readthedocs.io/en/latest/maven.html

18 http://h3xstream.github.io/find-sec-bugs/

OWASP Zed Attack Proxy Project (ZAP)

OWASP Zed Attack Proxy (ZAP)[19], is an open-source web application security scanner[20] sponsored by the OWASP foundation and release under the Apache 2.0 license. Some of the built-in features include an intercepting proxy server, traditional and AJAX[21] web crawlers, an automated scanner, and a fuzzer. It has a plugin-based architecture and an online 'marketplace'[22] which allows new or updated features to be added to extend the functionalities of ZAP. Since E-CORRIDOR will be a micro-services-based framework, ZAP provides a plug-in[23] to test RESTful web services compliant with OpenAPI[24] standard.

Note: Due to the E-CORRIDOR partners past experience, there are not many open source tools that support dynamic application security testing and that is why we selected only OWASP ZAP which is the most mature and advanced currently available.

4.2. Test Bed Environment Requirements

4.2.1. Test Bed Environment Infrastructure Configuration

The Test Bed Environment will consist of a number of Virtual Machines and devices that will run in IIT CNR Data Center.

E-CORRIDOR – Test Bed Environment Configuration		
Number of Machines	**Aim**	**Features**
1 Virtual Machine	This machine will host the latest development versions (for testing purposes) of all the containers implementing the E-CORRIDOR framework	4 CPU Cores 16 GB RAM 1 TB Storage UBUNTU server 20.04 LTS OS
1 Virtual Machine	This machine will host the latest development versions (for testing purposes) of the environment of the AT pilot	4 CPU Cores 8 GB RAM 200 GB Storage UBUNTU server 20.04 LTS OS
1 Virtual Machine	This machine will host the latest development versions (for testing purposes) of the environment of the S2C pilot	4 CPU Cores 8 GB RAM 200 GB Storage

19 https://www.zaproxy.org/

20 https://en.wikipedia.org/wiki/Web_application_security_scanner

21 http://www.adaptivepath.org/ideas/ajax-new-approach-web-applications/

22 https://www.zaproxy.org/addons/

23 https://www.zaproxy.org/docs/desktop/addons/openapi-support/

24 https://www.openapis.org/

		UBUNTU server 20.04 LTS OS
1 Virtual Machine	This machine will host the latest development versions (for testing purposes) of the environment of the ISAC pilot	4 CPU Cores 8 GB RAM 200 GB Storage UBUNTU server 20.04 LTS OS
1 Raspberry PI	This device will be used by the pilots to emulate sensors or devices installed in vehicles, trains or planes	4 GB RAM 32 GB Storage Raspberry PI OS
1 Mobile device	This device will be used by the pilots to emulate sensor or devices installed in vehicles, trains or planes	4 GB RAM 32 GB Storage Android 10 OS
At least 1 Automotive embedded device	This device will act an Electronic Control Unit and will be used by the pilots to emulate a working ECU inside the in-vehicle network for instance to generate and/or collect CAN bus data	32-Bit Single-Chip Microcontroller High Speed CAN Transceivers
1 Automotive In-Vehicle Infotainment System	This device will be used by the pilots to emulate a device in which vehicle data could be collected and shared with the E-CORRIDOR framework	1 GB RAM 32 GB Storage Android OS > 4.2

4.3. Production Environment Requirements

4.3.1. Production Environment Infrastructure Configuration

The Production Environment will consist of a number of Virtual Machines that will run in IIT CNR Data Center, and of a number of devices specific to the E-CORRIDOR pilots that will run in pilot owners' premises.

E-CORRIDOR – Production Environment Configuration		
Number of Machines	**Aim**	**Features**
1 Virtual Machine	This machine will host the latest stable versions of all the containers implementing the E-CORRIDOR framework	4 CPU Cores 16 GB RAM 1 TB Storage UBUNTU server 20.04 LTS OS
1 Virtual Machine	This machine will host the latest stable version of the environment of the AT pilot	4 CPU Cores 8 GB RAM 200 GB Storage

		UBUNTU server 20.04 LTS OS
1 Virtual Machine	This machine will host the latest stable version of the environment of the S2C pilot	4 CPU Cores 8 GB RAM 200 GB Storage UBUNTU server 20.04 LTS OS
1 Virtual Machine	This machine will host the latest stable version of the environment of the ISAC pilot	4 CPU Cores 8 GB RAM 200 GB Storage UBUNTU server 20.04 LTS OS
At least 1 Automotive embedded device	This device will act an Electronic Control Unit and will be used by the pilots to emulate a working ECU inside the in-vehicle network for instance to generate and/or collect CAN bus data	32-Bit Single-Chip Microcontroller High Speed CAN Transceivers
1 Automotive In-Vehicle Infotainment System	This device will be used by the pilots to emulate a device in which vehicle data could be collected and shared with the E-CORRIDOR framework	1 GB RAM 32 GB Storage Android OS > 4.2

4.4. Pilot Requirements

The following sections report the list of requirements needed for the pilot infrastructures used as test bed and the integration activities, production, as well as specific software requirements.

4.4.1. Pilot AT

4.4.1.1. Test Bed & Production requirements

Table 13 – Pilot AT – Test Bed & Production Requirements

ID	Requirement	Priority	Description
E-CORRIDOR-Tst-AT-01	VM for analytics	MUST	Pilot AT needs a VM with more than 16GB of RAM potentially with the change of a discrete GPU (Graphical Processing unit) to accelerate the machine learning (ML) algorithms
E-CORRIDOR-Tst-AT-02	IP Camera	COULD	IP connected camera to identify passengers and identify security event
E-CORRIDOR-Tst-AT-03	LiDAR camera	COULD	Light Detection and Ranging camera to identify passengers and identify security event

E-CORRIDOR-Tst-AT-04	Bluetooth beacons	SHOULD	To monitor the passenger flow
E-CORRIDOR-Tst-AT-05	WiFi antenna	SHOULD	To monitor the passenger flow
E-CORRIDOR-Tst-AT-06	Mobile devices (smartphone)	SHOULD	For AT-US-05 and all the other user stories related to passenger identification
E-CORRIDOR-Tst-AT-07	Intel RealSense camera, Raspberry Pi 3 with camera	SHOULD	Alternative solution to LiDAR and IP camera for passenger identification

4.4.1.2. Integration Requirements

Table 14 – Pilot AT – Integration Requirements

ID	Requirement	Priority	Description
E-CORRIDOR-Tst-Int-AT-01	Confidentiality	MUST	Support an integration protocol that assure confidentiality and data integrity
E-CORRIDOR-Tst-Int-AT-02	Jetson AI-Computer Emulator	COULD	Virtual live camera emulator
E-CORRIDOR-Tst-Int-AT-03	Docker engine	MUST	Container management and provisioning service
E-CORRIDOR-Tst-Int-AT-04	Rancher	SHOULD	Micro-service orchestration

4.4.1.3. Software Requirements

Table 15 – Pilot AT – Software Requirements

ID	Requirement	Priority	Description
E-CORRIDOR-Tst-Sw-AT-01	Programming Language	MUST	The code will be written in Python or compiled
E-CORRIDOR-Tst-Sw-AT-02	Apache Drill, Kibana	SHOULD	Data query and visualization
E-CORRIDOR-Tst-Sw-AT-03	PySyft	COULD	Federated learning library
E-CORRIDOR-Tst-Sw-AT-04	KeyCloak, Aerobase, Apereo CAS	COULD	Open source identity and access management
E-CORRIDOR-Tst-Sw-AT-05	Traefik	SHOULD	Proxy and load balancer for micro-services
E-CORRIDOR-Tst-Sw-AT-06	Istio	SHOULD	Secure connection, routing and monitoring of micro-services

E-CORRIDOR-Tst-Sw-AT-07	Mosquitto	SHOULD	MQTT broker, client service to share IoT data
E-CORRIDOR-Tst-Sw-AT-08	Kafka	SHOULD	Pub/sub messaging service

4.4.2. Pilot S2C

4.4.2.1. Test Bed & Production requirements

Table 16 – Pilot S2C – Test Bed & Production Requirements

ID	Requirement	Priority	Description
E-CORRIDOR-Tst-S2C-01	VM for S2C pilot	MUST	Pilot S2C needs a VM for running parts of the E-CORRIDOR framework (capable of running the FHE if needed)
E-CORRIDOR-Tst-S2C-02	Device that is compatible with OBD (or CAN BUS) for monitoring and sending GPS and driving behaviour data.	SHOULD	Needed for the driver identification analytics (WP7)
E-CORRIDOR-Tst-S2C-03	Data storage	MUST	Data storage infrastructure for the shared data

4.4.2.2. Integration Requirements

Table 17 – Pilot S2C – Integration Requirements

ID	Requirement	Priority	Description
E-CORRIDOR-Tst-Int-S2C-01	Confidentiality	MUST	Support an integration protocol that assure confidentiality and data integrity
E-CORRIDOR-Tst-Int-S2C-02	API integrations	MUST	API integrations between the S2C partners information system and the E-CORRIDOR framework via RESTful mechanism and OpenAPI specification

4.4.2.3. Software Requirements

None at this stage.

4.4.3. Pilot ISAC

4.4.3.1. Test Bed & Production requirements

Table 18 – Pilot ISAC – Test Bed & Production Requirements

ID	Requirement	Priority	Description
E-CORRIDOR-Tst-ISAC-01	At least 1 Automotive embedded device	MUST	This device will act an Electronic Control Unit and will be used by the ISAC pilot to emulate a working ECU inside the in-vehicle network to generate, collect, share and analyse CAN bus data (ISAC-US-02, ISAC-US-06, ISAC-US-07)
E-CORRIDOR-Tst-ISAC-02	1 Automotive In-Vehicle Infotainment System	MUST	This device will be used by the ISAC pilots to emulate a device in which vehicle data could be collected and shared with the ISAC-MMT
E-CORRIDOR-Tst-ISAC-03	At least 5 Raspberry and 2 NVIDIA Jetson with a sensors kit (camera / proximity / temperature / humidity / air pressure / Gas)	SHOULD	These devices will be used to emulate an airport IoT network scenario in which network information could be collected, analysed and shared with the ISAC-MMT
E-CORRIDOR-Tst-ISAC-04	2 VMs for analytics and frontend platform.	MUST	This 2 VM swill host the backend (for the analytics) and the frontend platform (analysis results and representation) of the ISAC pilot. Pilot ISAC, needs a VM with more than 16GB of RAM potentially with a GTX NVIDIA GPU for the machine learning activities and a VM for the frontend part.

4.4.3.2. Integration Requirements

Table 19 – Pilot ISAC – Integration Requirements

ID	Requirement	Priority	Description
E-CORRIDOR-Tst-Int-ISAC-01	Confidentiality	MUST	Support an integration protocol that assure confidentiality and data integrity.
E-CORRIDOR-Tst-Int-ISAC-02	Authenticity	MUST	Support an intrusion protection system able to authenticate the ECU in an intra-vehicle network when it aims at sending cross partition CAN frame

E-CORRIDOR-Tst-Int-ISAC-03	Authenticity, Confidentiality, Integrity	MUST	Support an AUTOSAR compliant secure on-board communication module to guarantee all security properties on an intra-vehicle CAN-bus communication

4.4.3.3. Software Requirements

Table 20 – Pilot ISAC – Software Requirements

ID	Requirement	Priority	Description
E-CORRIDOR-Tst-Sw-ISAC-01	Programming Language	MUST	C, Java, Python, JavaScript
E-CORRIDOR-Tst-Sw-ISAC-02	Framework	MUST	Apache Drill, Kibana, Elasticsearch, Tensorflow, Flask
E-CORRIDOR-Tst-Sw-ISAC-03	Software tools	MUST	Wireshark

5. Conclusions

This document presented the list of E-CORRIDOR framework requirements coming from the project pilots' analysis and synthesis, considering both the functional and non-functional aspects. Requirements consider needs about key aspects for the framework, like the data sharing and analysis necessities, but also evaluate security and quality facets the E-CORRIDOR framework must implement.

We also outlined the infrastructure that will be used to develop and operate the E-CORRIDOR reference implementation of the framework, considering both the tools needed, the computing resources and the pilot needs.

Next WP5 milestone (MS2) due at M12 will present a first version of the E-CORRIDOR overall architecture of the framework. The architecture will need to include outcomes from WP6 about the ISI and IAI infrastructures as well as WP7 about the data analytics techniques to be integrated into the IAI. WP8 will also receive such inputs to allow the integration of advanced authentication and authorisation security services using the designed E-CORRIDOR framework.

6. References

Here we provide bibliography references used in the document:

[1] Trabelsi S., Sendor J., Sticky policies for data control in the cloud. Proceedings of the 10-th Annual International Conference on Privacy, Security and Trust, pp.75-80, 2012.

[2] K. Brennan, A Guide to the Business Analysis Body of Knowledge, International Institute of Business Analysis, 2009.

[3] ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.

[4] Capability Maturity Model Integration (CMMI), CMMI Institute, ISACA, Carnegie Mellon University, 2018.

[5] Open Web Application Security Project Software Assurance Maturity Model, OWASP 2020.

[6] Build Security-In Maturity Model, BSIMMv11, 2020.

[7] OpenID Connect, OpenID Foundation (2014), https://openid.net/connect/ (fetched Nov 2020).

[8] The OAuth 2.0 Authorization Framework - RFC6749, Internet Engineering Task Force (IETF), D. Hardt, Ed., Microsoft, October 2012, https://tools.ietf.org/html/rfc6749 (fetched Nov 2020).

[9] Security Assertion Markup Language (SAML), v2.0 (2005), Security Services (SAML) Technical Committee, https://wiki.oasis-open.org/security/FrontPage (fetched Nov 2020).

[10] electronic IDentification, Authentication and trust Services (eIDAS), EU Regulation 910/2014, https://eur-lex.europa.eu/eli/reg/2014/910/oj (fetched Nov 2020).

[11] OpenAPI Specification, OpenAPI Initiative, 2020, http://spec.openapis.org/oas/v3.0.3 (fetched Nov 2020).

[12] VMware vSphere Infrastructure, VMware Inc., 2020, https://www.vmware.com/products/vsphere.html (fetched Nov 2020).