SEMIoTICS

Deliverable D2.3
Requirements specification of SEMIoTICS framework

Deliverable release date	Initial 31.07.2018, revised 25.11.2019
Authors	1. George Hatzivasilis, Nikolaos Petroulakis (FORTH) 2. Jordi Serra, Luis Sanabria-Russo, David Pubill, Angelos Antonopoulos, Christos Verikoukis (CTTC) 3. Kostas Fysarakis, Jason Somarakis (STS) 4. Domenico Presenza (ENG) 5. Danilo Pau, Mirko Falchetto (ST-I) 6. Tobias Marktscheffel (Uni Passau) 7. Łukasz Kempiński, Łukasz Kałużny, Łukasz Ciechomski, Sylwester Zieliński (BlueSoft) 8. Prodromos Vasileios Mekikis (IQUADRAT)
Responsible person	Danilo Pau, Mirko Falchetto (ST-I)
Reviewed by	Nikolaos Petroulakis (FORTH), Domenico Presenza (ENG), Vivek Kulkarni (SAG)
Approved by	PTC Members (Vivek Kulkarni, Nikolaos Petroulakis, Ermin Sakic, Mirko Falchetto, Domenico Presenza, Verikoukis Christos) PCC Members (Vivek Kulkarni, Ioannis Askoxylakis, Verikoukis Christos, Georgios Spanoudakis, Domenico Presenza, Danilo Pau, Joachim Posegga, Darek Dober, Kostas Ramantas, Ulrich Hansen)
Status of the Document	Final
Version	1.0 revised
Dissemination level	Public

Table of Contents

ACRONYMS TABLE

Acronym	Definition
ASIC	Application-Specific Integrated Circuit
CPU	Central Processing Unit
DoS	Denial of Service
DPDK	Data Plane Developer's Kit
FPGA	Field-Programmable Gate Array
GA	Grant Agreement
IHES	Intelligent Heterogeneous Embedded System
IoT	Internet of Things
IIoT	Industrial Internet of Things
KVM	Kernel-based Virtual Machine
LXD	Linux Containers
JSON	JavaScript Object Notation[1]
FW	Firmware
LWM2M	Lightweight Machine-to-M
mW	milliWatts
M2M	Machine-to-M
MCU	Micro Controller Unit
MQTT	Message Queuing Telemetry Transport[2]
NETCONF	Network Configuration Protocol
NFV	Network functions virtualization
NFVO	NFV orchestrator
OFCONF	OpenFlow Configuration
OVSDB	Open vSwitch Database Management Protocol
POP	Point of Presence
QoS	Quality of Service
RO	Resource Orchestrator
SDN	Software-Defined Networking
SoS	System of System
SARA	Socially Assistive Robotic Solution for Mild Cognitive Impairment or mild Alzheimer's disease
SEMIoTICS	Smart End-to-end Massive IoT Interoperability, Connectivity and Security
SLAM	Simultaneous Localization and Mapping
SPDI	Security, Privacy, Dependability, and Interoperability
SSD	Solid State Disk
UC	Use Case
VIM	Virtualized Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
vSwitch	Virtual Switch
VTN	Virtual Tenant Networks
WoT	Web of Things

[1] https://en.wikipedia.org/wiki/JSON
[2] https://en.wikipedia.org/wiki/MQTT

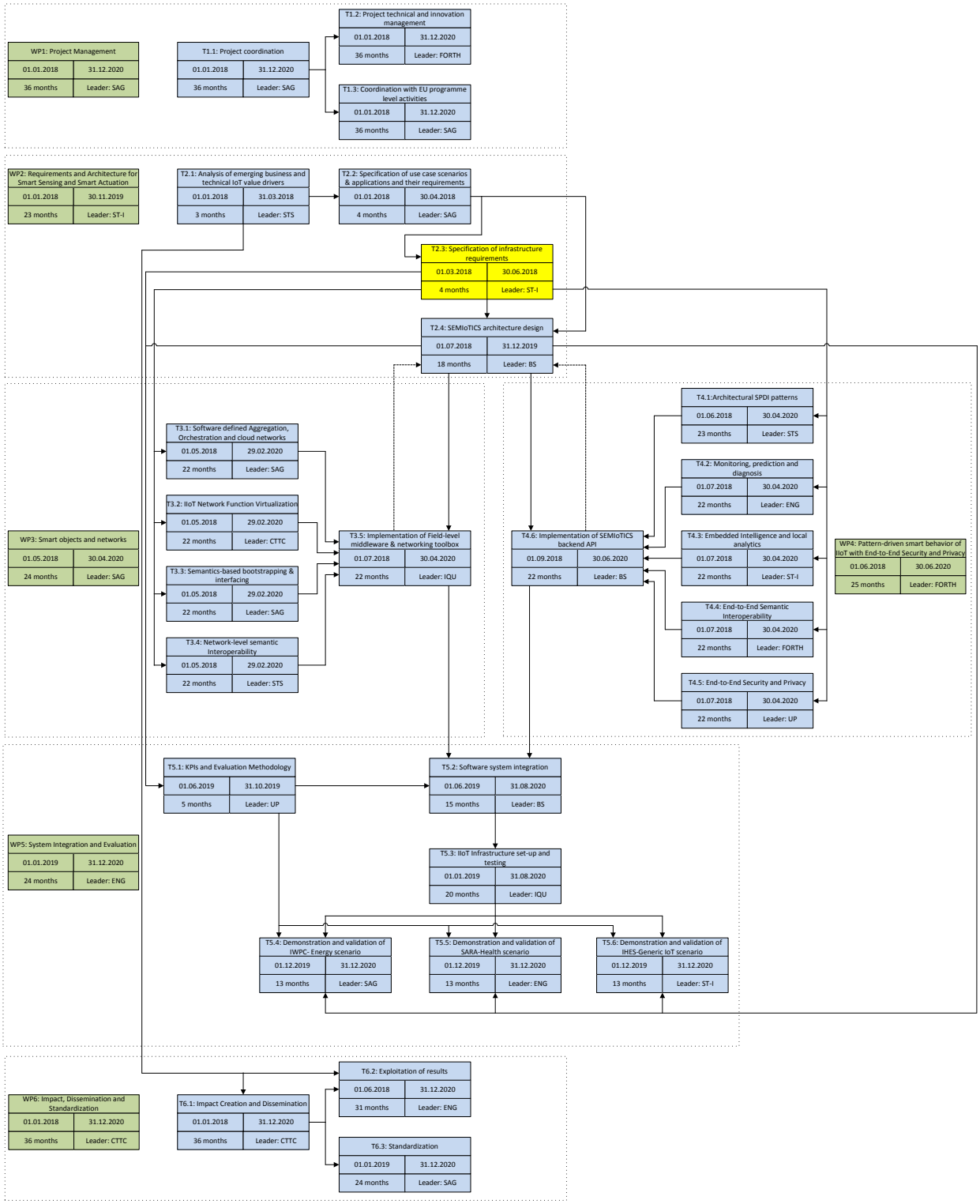
1 INTRODUCTION

This deliverable will provide the specification of the complete set of functional and non-functional requirements for the SEMIoTICS framework. These will cover all levels of its envisaged implementation stack, and will drive the development of components, techniques and algorithms in WP3 and WP4, and the integration in WP5.
Particular attention will be given on covering aspects regarding IoT security and privacy solutions, and the intelligence/analytics capabilities required at different levels of the framework implementation stack, including the IoT platform, SEMIoTICS framework, IoT applications, smart/intelligent objects/components level, up to the (logical) intra-level communication interfaces envisaged.

1.1 SEMIoTICS approach

The objective of this document is to present the first draft of the SEMIoTICS high level and the overall system architecture. The deliverable is associated with the deliverable D2.2 where the SEMIoTICS application in 10 use cases is detailed. This version is also linked with the upcoming deliverables for the WP3, WP4, and WP5, where the presented content will be utilized.
Chapter 2 describes the general requirements for the SEMIoTICS platform and infrastructure. The various technological aspects are determined, including functional and non-functional operations. Moreover, the IoT security and privacy design aspects and mechanisms are defined here.
Chapter 3 highlights the domain specific features of the three SEMIoTICS use cases (wind energy, e-health, smart sensing). The aforementioned infrastructure requirements of the previous chapter are now mapped in each specific scenario.
Finally, Chapter 4 concludes and links this document with the related upcoming deliverables for WP3, WP4, and WP5. Furthermore, the Appendix A summarizes the requirements for the 10 use cases, which are detailed in the deliverable D2.2. These requirements are mow mapped to the related system aspects that are presented in Chapter 2 of this deliverable.

1.2 PERT chart of SEMIoTICS



Please note that the PERT chart is kept on task level for better readability.

2 SEMIOTICS INFRASTRUCTURE REQUIREMENTS

The final requirements set reported in this deliverable have been identify within the SEMIoTICS consortium considering several aspects and detailing them through several cycles iterations done during Task 2.3 activities. The methodology used considered the set of initial generic requirements expressed in D2.2 (see Appendix A) on one side and the generic SEMIoTICS architecture proposed in the Grant Agreement on the other side to identify the final set of requirements (mostly functional) for each level of the architecture and for each one of the three main use cases considered in SEMIoTICS. This final set will be used to derive Architectural Requirements. In this section 2 the requirements are identified according to the proposed architecture, whereas in section 3 they are clustered for each specific UC supported by the architecture, in order to proper define the tasks and development phases in WP4 for these three considered scenarios.
Also, to the definition of these requirements contributes the main outcomes from D2.1 related to enabling technologies, and, as said, considering the SEMIoTICS ecosystem based on the layered infrastructure declared in the Grant Agreement. They were identified by experts in the field, for the different technologies considered, considering regulations (as in the case of the security and privacy ones) and specific needs of the use cases and UCs stakeholders (e.g. Wind Park Operators for UC1, Patients for UC2, IoT Operators, technology implementers for UC3, etc.)
A specific naming convention has been adopted and a set of requirements tables are provided in order to have a coherent description of the requirements throughout the document. In particular, a requirement has the following dot annotated signature:

- **R.<req_type>.<req_id>** where initial **R** stands for “Requirement”, **<req_type>** identify the type of requirement clustered on one of these categories: **BC**=”Backend Cloud”, **GP**=”General Platform”, **NL**=”Network Level”, **S**=”Security”, **P**=”Privacy”, GSP=”Generic Security and Privacy”, **UCx**=”Use Case #x” where x could be either 1, 2 or 3 and identify the specific use case as described in D2.2. Finally, the **<req_id>** is a number that enumerates the requirement.

Moreover, each requirement table in this document is defined by five columns “Req-ID”, “Former Req-ID”, “Functional”, “Description”, “Req. level” where:

- **Req-ID** is the column containing the consolidated requirement set for SEMIoTICS architecture definition, clustered by an ID expressed by above mentioned naming convention.
- **Former Req-ID** is an optional table that refers (if present) to the initial generic requirements that were initially defined during the use case scenarios definition (and that are mainly part of the deliverable D2.2.
  See Annex Table A for a complete overview of former requirements identified during the initial phase of the project). When the requirement has been identified, introduced during task 2.3 activities a reference “new” is added. Finally, if the requirement has been further elaborated or characterized from former initial requirements an “Derived from RX.Y” label has been provided.
- **Functional** column is of type Boolean (True/False) reporting if given requirement if a functional or non-functional requirement.
- **Description** is a textual description of the requirement.
- **Req. level** column indicates the level of the requirement according to the RFC 2119 – IETF, https://www.ietf.org/rfc/rfc2119.txt standard convention.

This document aims at identifying the complete set of requirements that will allow SEMIoTICS consortium to provide at the end of the project the eco-system able to support the functionalities of the three main use cases declared in D2.2, whereas the generic ones has been kept considering in the picture also the ones considered in the Non-SEMIoTICS ones, according to the definition provided in D2.2.

2.1 General Platform/technological/infrastructure Requirements

SEMIoTICS must provide end-to-end connectivity between the heterogeneous IoT devices, at the field level, and the heterogeneous IoT Platforms at the backend cloud level. The overall infrastructure must take into account the scalability requirement due to the fast-paced growth of IoT devices. Moreover, the heterogeneous IoT devices and IIoT applications lead to different quality of service (QoS) connectivity requirements. That is, some of them need a strict low-latency and reliable communication, others require a reliable connection, and some of them may not require neither low latency nor a reliable communication. Thereby, SEMIoTICS architecture requires a high adaptation capability to accommodate different QoS

connectivity needs in a scenario with a massive number of IoT devices. This adaptability requirement affects all the SEMIoTICS layers. Namely, at the upper layer, the IoT applications can detect events requiring a QoS change in terms of e.g. latency or reliability. As a consequence, the IIoT application will trigger a network reconfiguration need, through an SPDI pattern, which is generated in a middleware between the IIoT application and the SDN controller, at the IoT backend cloud. Then, the IoT backend cloud and the SDN controller interact through northbound software interfaces. Afterwards, the SDN controller reconfigures the network resources to guarantee the QoS requirements triggered by the IIoT application. To do so, it interacts through southbound software interfaces with the network nodes, e.g. switches, routers or IoT Gateways. Finally, the IoT devices can establish an end-to-end connection with the IIoT applications with the proper QoS requirement. Furthermore, the SDN controller, can give feedback to the future generation of SPDI patterns. Namely, the SDN controller gathers control information regarding the state of the network, which feeds the IoT backend middleware blocks dealing with the monitoring, control, adaptation and learning. This information will serve to monitor and analyse the effectiveness in the application of the patterns. Thereby, if failures or deficiencies in the application of the patterns are detected, one can avoid using the same patterns in the future or learn the required modifications of the patterns.

TABLE 1: GENERAL PLATFORM REQUIREMENTS

Req-ID	Former Req-ID	Functional	Description	Req. level
R.GP.1	Derived from R1.1	Yes	End-to-end connectivity between the heterogeneous IoT devices (at the field level) and the heterogeneous IoT Platforms (at the backend cloud level)	MUST
R.GP.2	Derived from UCs connectivity requirements	Yes	Scalable infrastructure due to the fast-paced growth of IoT devices	MUST
R.GP.3	Derived from UCs connectivity requirements	Yes	High adaptation capability to accommodate different QoS connectivity needs (e.g. low latency, reliable communication)	MUST
R.GP.4	Derived from UCs connectivity requirements	Yes	Detection of events requiring a QoS change and triggering network reconfiguration need by SPDI pattern	MUST
R.GP.5	Derived from UCs connectivity requirements	Yes	Interaction between SDN controller and IoT backend cloud through a dedicated interface (called northbound software interface)	MUST
R.GP.6	Derived from UCs connectivity requirements	Yes	Interaction between SDN controller and network nodes (e.g. switches, routers or IoT Gateways) through dedicated interface (called southbound software interface)	MUST
R.GP.7	Derived from UCs connectivity requirements	Yes	SDN controller giving feedback for a future generation of SPDI patterns to avoid using the same pattern in case of failure	MUST

2.2 Backend/Cloud Level

The SEMIoTICS backend cloud contains a set of heterogeneous IoT platforms, e.g. FIWARE or MindSphere. Connectivity between any of those IoT platforms and the SEMIoTICS network must be ensured regardless of the heterogeneity. In order to satisfy this requirement, software interfaces must be

specified, in the SDN jargon this is the so-called northbound software interfaces. These software interfaces will be compliant with the SEMIoTICS SDPI patterns specifying composition structures for integrating smart objects and components of IoT enabling platforms. Regarding the SDPI patterns and the network level, the IoT backend cloud has to define patterns specifying adaptation actions for the network. That is, adaptation actions for the network can be concluded given the monitoring of the network state, also the connectivity and the security requirements. Examples of these adaptation actions are redefining routing paths or establishing a low latency and reliable connection to attend an event generated by an IoT device.

The need to support multiple IoT platforms creates many requirements for component infrastructure, some functions need to be separated and shared among all connected platforms. On the other hand, SEMIoTICS platform should have access to functions available only to selected IoT platforms.

Monitoring

Monitoring will be created as separate component, this way all the logged events/messages can be saved/accessed/viewed in one unified way.

Security

Security, especially authentication and authorization part, should be provided by one dedicated component integrated with SEMIoTICS API. From field device perspective all functions should be provided by one unified API, the filed device itself shouldn't be aware which platform it is connecting to.

Message routing

Using multiple IoT platforms means more out of the box functions to start with. To be able to utilize all out of the box functions special router component should be created. The main role of the component will be sending messages to different IoT platforms for further processing. The message will be routed based on messages types/sources/etc.

Connectivity

Connectors to all used IoT platforms need to be developed. They will be used to connect backend with IoT platforms. Connecting new IoT platform to SEMIoTICS solution should only require developing a new connector, further tasks related to platform should be a configuration only.

Each of components mentioned above will be obtained in one of two ways: created from scratch or one of ready to use components available on selected IoT platform will be used and made available for all of the other IoT platforms.

Cloud computing has a very important role in SEMIoTICS project, facilitating the transition from dedicated "black box" hardware to software-defined solutions. Cloud computing solutions include public offerings such as Amazon Web Services (AWS), Google Compute Engine, and Microsoft Azure, or private clouds deployed in-house through software solutions provided by leading vendors such as VMWare as well as Open Source software solutions. Deploying functionalities in the remote cloud only, cannot scale in each and every vertical use case as in medium to large IoT deployments, targeted by SEMIoTICS. Hence, the SEMIoTICS Backend Cloud will consider local and remote cloud computing solutions that will allow monitoring, caching, security and data analytics functions to be virtualized and placed in the local or remote clouds, or event directly at IoT smart objects and Field level IoT gateways, subject to resource availability and KPI constraints. The SEMIoTICS Backend Cloud solution, shown in Figure 1**,** will be underpinned by the ETSI NFV reference architecture. Using NFV, virtual Network Functions can be run as software instances on top of an abstracted infrastructure for a more dynamic and cost-effective network sharing.

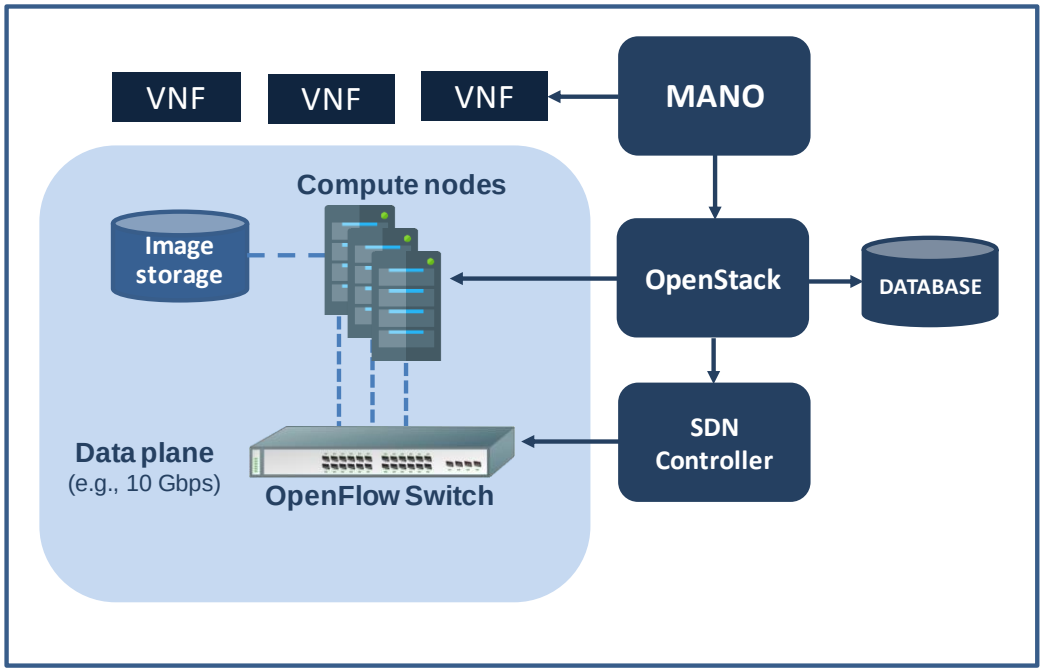


FIGURE 1: BACKEND CLOUD ARCHITECTURE

The following technologies and architectural components will be employed as part of the SEMIoTICS SDN/NFV backend cloud. Their functional requirements are detailed in the Table that follows:

- **Controller Node**
  - The Controller Node hosts the SDN and MANO controllers that are responsible for Network Orchestration, as well as the on boarding, placement, and lifecycle management of SEMIoTICS services.
  - Each controller service will be hosted in a separate Linux Container.

- **Hypervisor Nodes**
  - The SEMIoTICS backend cloud will include multiple Hypervisor nodes, that will host the Virtual Machines (VMs) and Virtual Network Functions (VNFs).

- **Virtualization Technology**
  - KVM (*Kernel-based Virtual Machine*) is currently the industry standard virtualization solution, which allows VMs to be natively executed at a host operating system.
  - Linux Containers (LXD) is an emerging virtualization solution which allows VMs to run almost to the "bare metal" with minimal performance penalties, but with the requirement that they share the same kernel with the host.

- **Virtual Tenant Network (Data Plane)**
  - Virtual Networking techniques will be employed for the VM tenant networks. A Virtual Switch (vSwitch) solution will be employed, which must be OpenFlow enabled so it can be dynamically configured by an SDN Controller. Performance optimization frameworks such as compiling *Intel's DPDK* (*Data Plane Developer's Kit*) will also be investigated.
  - The vSwitch solution provides Layer2 connectivity to the Hypervisor nodes that are part of the same tenant network, while a virtual Router (vRouter) handles Layer3 routing among tenant networks.

- **SDN Controller**
  - An SDN controller provides centralized control of the Virtual Network and enables the deployment of Network Slices at the VM tenant network. Modern VIMs already support some form of centralized network management and control (e.g., OpenStack Neutron) but

without the flexibility and comprehensive APIs of dedicated SDN controllers (e.g., OpenDaylight, ONOS, etc.).

- **Virtualized Infrastructure Manager (VIM)**
  - A VIM framework will be employed to manage the Cloud Computing environment, deploying VMs to the hypervisors and managing their lifecycle. Modern VIMs, such as OpenStack, are complex software frameworks with multiple components that handle security and authentication, VM image storage, VM instantiation and termination, etc.

- **Management and Orchestration (MANO) Framework**
  - In a typical Management and Orchestration (MANO) framework, the Network Service Orchestrator (NSO) handles the delivery of end-to-end network services, interacting with the Resource Orchestrator component. It supports the lifecycle management of network services, catalogue management and on-boarding/configuration of network services and VNFs.
  - A resource orchestrator (RO) coordinates the allocation and setup of the computing, storage and network resources that are necessary for the instantiation and interconnection of VNFs. For this purpose, the RO may interact with multiple Virtualized Infrastructure Managers (VIMs).
  - The NFV orchestrator (NFVO) is responsible for orchestrating VNFs across Multiple VIMs and Multiple Sites (POPs). Moreover, it performs VIM Resource Checks and Resource Allocation to ensure that the VNF requirements are met. Several open source orchestrators will be investigated such as *OpenStack's Tacker*, *ETSI's Open Source Mano (OSM)* and *Open-Baton*.

TABLE 2: BACKEND/CLOUD LAYER REQUIREMENTS[3]

Req-ID	Former Req-ID	Functional	Description	Req. level[4]
R.BC.1	New	Yes	Controller Node requirement: At least 6 CPU cores and 32 GB RAM	MUST
R.BC.2	New	Yes	Controller Node requirement: At least 2 Network interfaces	SHOULD
R.BC.3	New	Yes	Controller Node Requirement: Linux OS	MUST
R.BC.4	New	Yes	Controller Node Requirement: Solid State Disk (SSD) of at least 256 GB	MUST
R.BC.5	New	Yes	Hypervisor Requirement: At least 4 CPU cores and 8 GB RAM	MUST
R.BC.6	New	Yes	Hypervisor Requirement: At least 2 Network interfaces	SHOULD
R.BC.7	New	Yes	Hypervisor Requirement: Virtualization Extensions (Intel VT-x/AMD-V) must be supported by the Hypervisor CPU for hardware acceleration of VMs.	MUST
R.BC.8	New	Yes	Hypervisor Requirement: KVM must be supported by the Hypervisor Linux OS	MUST

[3] These backend requirements have been identified by expert in the fields of SEMIoTICS consortium considering the use case scenarios descripted on D2.2 for the three main SEMIoTICS use cases. Also specific HW requirements has been derived from the requirements for the SW platform ecosystem that will be used by SEMIoTICS.
[4] According to: RFC 2119 – IETF, https://www.ietf.org/rfc/rfc2119.txt

R.BC.9	New	Yes	Hypervisor Requirement: Linux Containers (LXD) must be supported by the Linux OS	MUST
R.BC.10	New	Yes	Virtual Switch requirement: Support for OpenFlow protocol	MUST
R.BC.11	New	Yes	Virtual Network requirement: Support for GRE, VLAN, and VXLAN tunnels for virtual tenant networking.	MUST
R.BC.12	New	Yes	The VIM and Virtual Network frameworks must support Interfaces that enable VM tenant networking	MUST
R.BC.13	New	Yes	Interface between the VIM and the SDN controller to allow Tenant Network Slicing	MUST
R.BC.14	New	Yes	Interfaces among the MANO entities (NFO, RO, NFVO) and the VIM must ensure seamless interoperability among different entities of the Backend Cloud	MUST
R.BC.15	Derived from generic security requirements for the UCs	Yes	Secure communication among the various Backend Cloud components (e.g., use of dedicated management network, appropriate Firewall rules)	MUST
R.BC.16	New	Yes	Prediction mechanism based on the data stored in the cloud	MUST
R.BC.17	New	Yes	Quantitative/statistical analysis of the cloud data	MUST
R.BC.18	Derived from UCs requirements on Local Analytics	Yes	The backend layer must feature SPDI pattern reasoning embedded intelligence capabilities	MUST
R.BC.19	New	Yes	The backend layer should feature pattern-driven cross-layer orchestration capabilities	SHOULD
R.BC.20	New	Yes	The backend layer must aggregate intra-layer as well as inter-layer SPDI status information to enable local and global intelligence reasoning and adaptation	MUST

2.3 Network Level

The main functionality of the network is to provide the connectivity between the multiple IoT Gateways and the IoT backend cloud. In this regard, the scalability is an important requirement for the network, as it must support traffic generated by a fast-growing number of IoT smart objects, IoT backend users, and applications. Moreover, IoT is characterized by heterogeneous devices, IoT backend cloud platforms, and applications. Thereby, the network must comply with different communications requirements in terms of e.g. latency and reliability demanded by the IoT devices and applications at hand. Also, it must provide interfaces that permit the interaction with a great number of heterogeneous IoT platforms.

A requirement related to the heterogeneity is the dynamicity or adaptability of the network. Namely, IoT data in many cases are event driven. Thereby, the network must be flexible and dynamic in order to allocate resources to spontaneous events requiring low end-to-end latency and a reliable communication between the IoT Gateways and the IoT backend. Another important requirement that the network must support is the end-to-end security and privacy. For example, in the SEMIoTICS use case that deals with the renewable energy (subsection 3.1), security attacks can occur at the network level to manipulate the information transmitted between the Wind turbines and the control centre.

SDN and NFV are the cornerstone to support the requirements mentioned in the previous paragraph. Namely, in the SDN approach, the control and data planes are separated, and the network management is based on a centralized approach. Thereby, the SDN controller has a global view of the network state (e.g. links' traffic load, reliability and latency per link). This information is in turn made available to applications, such as the NFV orchestrator or heterogeneous IoT backend platforms, through standard northbound interfaces. Moreover, network equipment such as switches or routers, can be dynamically reconfigured via software interfaces, so-called southbound interfaces, depending on the network state and the communication requirements. Thereby, this softwarization of the SDN approach eases the management of heterogeneous IoT devices which require different traffic guarantees. Namely, a functional requirement for SDN is to provide southbound interfaces with switches, IoT gateways and routers (e.g. OpenFlow, NETCONF, OFCONF, OVSDB). This dynamically sets routing paths and physical isolation of the traffic that guarantee the end-to-end latency and reliability requirements demanded by a fast-growing number of heterogeneous IoT devices and applications.

SDN paves the way to fulfil the network security and privacy requirements that are mentioned above. More specifically, SDN eases the detection of anomalies in the network, provided that it fulfils the requirements of having a global view of the network state, the separation of the control and data planes, and the centralized management of the network. As an example, the global view of the network state, through network traffic measurements, permits the detection of traffic anomalies that can span several network links. That is to say, it is easier to detect the origin and destination of the anomalous flow of information and when this anomaly is occurring. That detection would be difficult just having a local view of the network. Moreover, network metrics provided via northbound interfaces can be used by Virtual Functions (VF) orchestrators, such as OSM, in order to spin such VFs at computing nodes satisfying IoT applications' requirements regarding latency and reliability of the communication.

To realize an SDN/NFV deployment such as the envisioned for SEMIoTICS, forwarding elements or data paths, should support an OpenFlow (v1.3 or greater) agent. This will yield complete control of data paths' forwarding decisions to the SDN Controller, which may dictate forwarding rules based not only on source and destination of the packets, but also on estimated network congestion, traffic priority, or other criteria. Moreover, in order to enable the spinoff of VFs, data paths must be capable of running Virtual Machines (VM) or containerized VFs using Docker, or Linux Containers (LXC). This coupling of network metrics (obtained from the SDN Controller via northbound interfaces) and computing nodes allows a Management and Orchestration software (i.e. OSM) to spin VFs closer to where they are requested, leading to potential reduction in latency.

Based on the requirements detailed above, data paths must be able to run a software switch like Open vSwitch, a popular production-ready virtual switch with an OpenFlow agent supported by most Linux distributions. In turn, the SDN Controller, such as OpenDaylight, may use southbound interfaces to configure such data paths (OVSDB, or NET-CONF), and dictate forwarding rules via a control plane using OpenFlow. Additionally, as data paths are expected to work also as computing nodes for spinning VFs, they must be equipped with enough compute, memory, and storage resources. Embedded devices such as the Raspberry Pi 3 Model B+ may take up the role of a data path but may perform poorly as a computing

node due to its constrained resources. This is also true for other embedded devices such as the ODROID-XU4. A viable solution for boosting performance and number of VF per data path/compute node, is to spin VF in the form of containers. Otherwise, it is suggested to employ x86 devices with enough hardware resources, such as Intel NUCs x86 devices.

Most of the architectural components and functionality at the Network layer share the requirements of the Backend/Cloud. Herein, their specific role at the Network level is described. The reader is also referred to the table at the end of this subsection for further details.

Architecture requirements:

- **Controller Node**
  - This node will host the SDN controller, as well as the Virtual Infrastructure Manager (VIM), and the Management and Orchestration (MANO) controller.
  - It is physically located at the Backend/Cloud level.

- **Data paths / Hypervisor Nodes for NFV**
  - Such devices compose the Network layer, and are equipped with enough processing, networking, and storage resources that allows them to support a wide range of Virtual Network Functions (VNF), such as **virtual Switches**, **virtual Routers**, **virtual Firewalls**, and **virtual Load Balancers**.

- **Physical network infrastructure**
  - Physical links among data paths must be in the order of Gigabits per second (Gbps). Ideally, connections among data paths should be greater or equal to 1Gbps Ethernet. This will bound the delay experienced by applications.

- **Virtualization Technology**
  - Kernel-based Virtual Machine (KVM) is the industry standard for virtualization with VMs. Such technology allows the spin-off of VNF at data paths.
  - Linux Containers (LXD) allows to run virtual applications and VNFs with minimal performance overhead on the host OS. Each controller deployed at the Controller Node runs in a separate LXD.

- **SDN Controller**
  - Running at the Controller Node inside an LXD, it has a centralized view of the whole network.
  - Provides tools for the construction of Virtual Tenant Networks (VTN) or slices, which will be employed alongside VNFs to provide different levels of reliability and security.
  - SDN Controllers such as OpenDaylight, provides an interface via a so-called Modular Layer 2 (ML2) plugin with Virtual Infrastructure Manager (VIM), such as OpenStack. This facilitates the control and management of network slices.
  - Exposes a northbound interface for SDN applications, delivering link-state information, data path's flow or port tables, among other metrics gathered via southbound protocols, such as OpenFlow >v1.3.

- **Virtualized Infrastructure Manager (VIM)**
  - Manages the computing and storage resources (at data path nodes) to create, maintain, and terminate VNF and/or virtualized applications.
  - Exposes an interface to the SDN Controller (via OpenStack Neutron) in order to ease the configuration and maintenance of VTNs.
- **Management and Orchestration (MANO)**
  - Makes sure that VNF's requirements are met before scheduling it.
  - Said requirements comprise computing resources, but also network information gathered via SDN Controller's northbound APIs.

TABLE 3: NETWORK LAYER REQUIREMENTS[5]

Req-ID	Former Req-ID	Functional	Description	Req. level
R.NL.1	New	Yes	Controller Node requirement: At least 6 CPU cores and 32 GB RAM	SHOULD
R.NL.2	New	Yes	Controller Node requirement: At least 2 Network interfaces	SHOULD
R.NL.3	New	Yes	Controller Node Requirement: Linux OS	MUST
R.NL.4	New	Yes	Controller Node Requirement: Solid State Disk (SSD) of at least 1 TB	SHOULD
R.NL.5	New	Yes	Data paths / Hypervisor Nodes Requirement: At least 4 CPU cores and 8 GB RAM, at least 2, 1Gbps Network interfaces, Virtualization Extensions (Intel VT-x/AMD-V) must be supported by the Hypervisor CPU for hardware acceleration of VMs.	MUST
R.NL.6	New	Yes	Data paths / Hypervisor Nodes: KVM and Linux Containers (LXD) must be supported by the Hypervisor Linux OS	MUST
R.NL.7	New	Yes	Virtual Switch requirement: Support for OpenFlow v1.3 protocol or greater	SHOULD
R.NL.8	New	Yes	The VIM and Virtual Network frameworks must support Interfaces that enable VM tenant networking	MUST
R.NL.9	New	Yes	Interface between the VIM and the SDN controller to allow VTN	MUST
R.NL.10	New	Yes	Interfaces among the MANO and the VIM must ensure seamless interoperability among different entities of the Backend Cloud	MUST
R.NL.11	New	Yes	Secure communication with the various Backend Cloud components (e.g., use of dedicated management network, appropriate Firewall rules), as well as the communication between VIM, SDN Controller, and MANO, with data paths acting as computing nodes for VNF spinoff.	MUST
R.NL.12	Derived from UCs requirements on Local Analytics	Yes	The network layer must feature SPDI pattern reasoning local embedded intelligence capabilities	MUST
R.NL.13	New	Yes	The network layer must aggregate intra-layer monitored information to enable local intelligence reasoning and adaptation	MUST

NFV-level intelligence through dynamic reconfiguration enablers

One of the main features of SEMIoTICS architecture is the ability to modify or reconfigure Network Services (NS). As thoroughly described in D3.2, NS are composed of virtual and physical network functions (VNFs, and PNFs, respectively) connected together via virtual or physical links. The specification of the properties of each element within a NS, i.e. VNFs and virtual links, are collected in ETSI-

[5] These network layer requirements have been identified by expert in the fields of SEMIoTICS consortium considering the use case scenarios descripted on D2.2 for the three main SEMIoTICS use cases and the target architecture defined in GA. Also specific HW requirements has been derived from the requirements for the SW platform ecosystem that will be used by SEMIoTICS.

standardized Network Service Descriptors (NSd), which in turn are composed of VNF descriptors (VNFd) and Virtual Link descriptors (VLd). Such descriptors are then on boarded to the NFV Orchestrator (NFVO), which then uses it as blueprint for realising the NS via API calls to the Virtualised Infrastructure Manager (VIM) (details on the NFV MANO architectural framework are contained in D3.2).

SEMIoTICS envisions two types of NS reconfiguration: 1) descriptor-based, and 2) live NS reconfiguration. The following provides insight into these two types, as well as their caveats and enablers.

- Descriptor-based NS reconfiguration: it implies the update of an on boarded or yet-to-onboard descriptor. Any authorised party interested on a modification of a service (e.g. based on a pattern) should perform the modification in the descriptor itself (written in YAML), and then onboard/update it at the NFVO previous orchestration. This type of reconfiguration is far reaching, meaning that it is virtually possible to modify all the elements of the NS.
    - A specific example of this type of reconfiguration relates to the adjustment of scale-out operations' thresholds. An external entity may decide to change the scale-out trigger from 80% of vCPU usage to 70% or determine that the maximum number of scaled-out instances should be 4 instead of 3.
- Live NS reconfiguration: this assumes a NS is already running on top of the NFV infrastructure. Updating a running NS via NFVO is limited to the change of collected metrics (this implies updating the corresponding VNFd). Nevertheless, leveraging VIM's APIs it is possible to change network-level QoS policies in real time[6]. Live modification of NS is limited to the available APIs at NFVO and VIM.

The SEMIoTICS NFV component deals with VNFs and their properties (e.g. vCPU, images, storage, placement, etc.) rather than with network properties (which are delegated to SEMIoTICS SDN Controller, see D3.1). In SEMIoTICS, it is expected that any reconfiguration of NS (be it descriptor-based or live) would be performed by the Global Pattern Orchestrator (or any other authorised Pattern enforcement engine) via the NFV Management and Orchestration (MANO) Operations/Business Support System endpoint (refer to Figure 1 in D3.2)[7].

All in all, intelligence at the NFV level tightly correlates with allowing authorized external elements to interact with NSd and in some specific instances with VIM's APIs. Therefore, requirements encompass connectivity among NFV MANO elements, as well as the exposure of endpoints to other authorised SEMIoTICS components. From Table 3, the specific requirements for this functionality are: R.NL.8, R.NL.9, R.NL.10 (OSS/BSS operations through Os-Ma-Nfvo endpoint, refer to Figure 1 in D3.2), and R.NL.11.

2.4 Field / Devices Level

Any apparatus of these is placed at the lowest level of the system of system (SoS) hierarchy and consists of field devices such as sensors, actuators and computing units.
A sensor is a device, which detects or measures a physical property and transmit it to a computing device. For example, it collects data on temperature, pressure, and so on, convert it to electrical signals, and relay it up to the closest computing unit.
A computing unit is a unit of execution that performs a job composed by tasks or steps. For example, it can be either a CPU, a MCU, a FPGA or a dedicated ASIC. That unit receives real time data from sensors and process them accordingly to the assigned job, producing results forwarded to other computing units through a communication network. A special type of field device can be a supervisory device that is a computing unit (e.g. CPU) that monitors the other devices (or be one of them) to facilitate their control and adjustment of various parameters at any point in time without relying on the far (in terms of latency) cloud feedbacks. For example, setting job targets, historical archiving (if local storage is available), and setting initialization, start and shutdown of jobs are functionalities assigned to the supervisory device.

[6] There are operations that would inevitably incur in down time (e.g. VM scale up/down), although some of these issues may be leveraged at the application level (e.g. using load balancers, replicas, etc.).
[7] It is also possible for authorized components to reach the VIM APIs for a greater set of operations, nevertheless, these should be carefully specified and managed in order to avoid security risks (e.g. misconfiguration of VIM, mismanagement of resources, etc.).

An actuator is a component of a field device that is responsible for moving and controlling a mechanism or system, for example by starting a cooler if the temperature increases above a certain threshold. Actuator connected to a computing unit, received commands to actuate. Any computing unit shall use less energy as possible within the constraint of real time processing of acquired sensor data. Real time constraint depends on the speed at which sensor produce data at its output (e.g. in the case of a temperature sensor 1Hz) and the computing units shall process all of them without avoiding skip some of them due to lack of enough computing resources (e.g. time, clock cycles, memory, etc.).

Edge computing paradigm in SEMIoTICS brings computation and data storage on the field devices, and this affects the whole architecture. Devices became smarter, implements self-adaptation mechanism and instead of collecting raw data and sending to the cloud for the computation, can make directly a deep analysis on this data. The local analytics performed by a field device is use case dependent but in general has the goal to detect events or pattern in collected data. Thanks to this new paradigm the sensor readings can remain at Field / Device Level and the cloud receive only the outcome of the processing. Messages became more complex but rich of useful information instead of simple raw data. This results in a different type of communication based on events: this new paradigm has clear benefit in bandwidth usage but require strong communication protocol and standard message formats to reach the interoperability between heterogeneous devices and the cloud. This scenario makes communication protocol and connection types a very important aspects to consider: field devices can be connected in different ways using standard protocols (e.g. 1 to 1 - HTTP, 1 to many - MQTT) and connection (wireless, wired). Depending on power constraints and specific use case a given protocol may be more suitable than others. The aspects to consider when defining a connection link layer and a protocol are (among others): requested bandwidth, power consumption budget, communication range, communication type (reliable/unreliable/point-to-point, etc.),need of synchronization and type of connection. All this observed features and requirements are generalized in the Field / Device Level requirements table below.

TABLE 4 FIELD DEVICES REQUIREMENTS[8]

Req-ID	Former Req-ID	Functional	Description	Req. level
R.FD.1	Derived from UCs specific requirements	Yes	Field devices SHOULD be able to get data from the environment through sensors (sensors).	SHOULD
R.FD.2	Derived from UCs specific requirements	Yes	Field devices SHOULD be able to process data in near real time (process units).	SHOULD
R.FD.3	Derived from UCs specific requirements	Yes	Field devices SHOULD be able to control (at least) a mechanism / system (actuators).	SHOULD
R.FD.4	Derived from UCs specific requirements	Yes	Field devices SHOULD use a global clock for time synchronization.	SHOULD
R.FD.5	Derived from UCs specific requirements	Yes	Field devices SHOULD be able to interact with SEMIoTICS IIoT/IoT gateway dedicated components	SHOULD
R.FD.6	Derived from UCs specific requirements	Yes	Field devices MUST interoperate using a standard communication protocol like Rest APIs, COAP, MQTT.	MUST
R.FD.7	Derived from UCs specific requirements	Yes	Field devices MUST use standardize interoperable message format (e.g. JSON, etc.).	MUST

[8] These field devices requirements have been identified by expert in the fields of SEMIoTICS consortium considering the use case scenarios descripted on D2.2 and their specific devices, connectivity capabilities, legacy middleware constraints, and new features supported by SEMIoTICS in this field (e.g. the local embedded analytics).

R.FD.8	Derived from UCs specific requirements	Yes	Field devices MUST support secure bootstrapping / registration protocol.	MUST
R.FD.9	Derived from UC1 specific requirements	Yes	Field devices MUST be able to communicate with the IIoT Gateway / other architectural components.	MUST
R.FD.10	Derived mainly from UC3 specific requirements	No	Field devices SHOULD minimize data traffic.	SHOULD
R.FD.11	Derived mainly from UC3 specific requirements	No	Field devices SHOULD minimize energy consumption.	SHOULD
R.FD.12	Derived from UCs specific requirements	Yes	Greenfield device is expected to expose its capability over a W3C Thing Description, which semantically describes field resources, and to be computationally powerful enough to run a node-wot servient (that exposes the TD).	MUST
R.FD.13	Derived mainly from UC1 specific requirements	Yes	Brownfield device is assumed to consist of a sensor/actuator and a controller (PLC). The controller is expected to expose capability of its sensor/actuator over a native brownfield protocol (without the need for IIoT Gateway to interact directly with them).	MUST
R.FD.14	Derived from UCs specific requirements	Yes	The field layer must feature SPDI pattern reasoning local embedded intelligence capabilities	MUST
R.FD.15	Derived mainly from UC3 specific requirements	Yes	The field layer must aggregate intra-layer monitored information to enable local intelligence reasoning and adaptation	MUST

2.5 Privacy security requirements

This subsection summarizes the security and privacy requirements for SEMIoTICS. This includes general protection aspects as well as specific functional and non-functional operations which are defined in specific use cases in D2.2.

Regarding the general protection mechanisms, we consider main security technologies for preserving the Confidentiality, Integrity, and Availability (CIA) principles and privacy techniques that safeguard the user's private data and comply with the General Data Protection Regulation (GDPR). Other methods for enhancing security and privacy in the IoT domain are also included. The next table presents the general IoT security and privacy requirements.

TABLE 5: IOT SECURITY AND PRIVACY REQUIREMENTS

Req-ID	Former Req-ID	Functional	Description	Req. level
R.S.1	New	Yes	The confidentiality of all network communication MUST be protected using state-of-the-art mechanisms.	MUST

R.S.2	New	Yes	Authentication and authorization of the stakeholders MUST be enforced by the Network controller, e.g. through access and role-based lists for different levels of function granularities (overlay, customized access to service, QoS manipulation, etc.)	MUST
R.S.3	New	No	Sensors SHALL be identifiable (e.g. by a TPM module/smartcard) and authenticated by the gateway.	SHALL
R.S.4	New	Yes	All components from gateway, via SDN Controller, to cloud platforms and their users MUST authenticate mutually.	MUST
R.S.5	New	Yes	Before sensitive data is being transmitted, the respective components SHALL be authenticated as defined by requirements R.S.3 and R.S.4	SHALL
R.S.6	New	No	Sensors SHALL be able to encrypt the data they generate, i.e. their CPU and memory SHALL be sufficient to perform these cryptographic operations.	SHALL
R.S.7	New	Yes	The negotiation interface of the SDN Controller SHALL be secure against network-based attacks	SHALL
R.S.8	New	No	The honeypot SHALL be a dedicated or virtual server.	SHALL
R.S.9	New	No	The honeypot SHALL run a Linux based, hardened operating system.	SHALL
R.S.10	New	No	The honeypot SHALL have hardware with sufficient computational capabilities (based on traffic).	SHALL
R.S.11	New	No	The honeypot MUST have adequate bandwidth capacity to provide sufficient traffic for logging.	MUST
R.S.12	New	No	The honeypot MUST have networking capabilities for redirecting traffic and mirroring ports	MUST
R.S.13	New	No	The honeypot SHALL be able to execute software for capturing the suspicious traffic for further processing.	SHALL
R.S.14	New	Yes	The honeypot SHALL provide a data repository and backend processing.	SHALL
R.S.15	New	No	The honeypot MUST run on a flexible SDN infrastructure (probably open source).	MUST
R.S.16	New	Yes	The honeypot system MUST act as a dummy SDN component (controller or switch).	MUST
R.S.17	New	Yes	There MUST be an interface between the network controller and the network administrators for the designation of the applications' permissions.	MUST
R.S.18	New	Yes	All network functions SHALL be mapped to application permissions	SHALL
R.S.19	New	Yes	Policy conflicts among the controller applications SHALL be identified and resolved.	SHALL
R.S.20	New	Yes	Cloud platforms MUST be protected by a firewall against network-based attacks.	MUST
R.P.1	New	Yes	The collection of raw data MUST be minimized.	MUST
R.P.2	New	Yes	The data volume that is collected or requested by an IoT application MUST be minimized (e.g. minimize sampling rate, amount of data, recording duration, different parameters).	MUST
R.P.3	New	Yes	Storage of data MUST be minimized.	MUST
R.P.4	New	Yes	A short data retention period MUST be enforced and maintaining data for longer than necessary avoided.	MUST

R.P.5	New	Yes	As much data as possible MUST be processed at the edge in order to hide data sources and not reveal user related information to adversaries (e.g. user's location).	MUST
R.P.6	New	Yes	Data MUST be anonymized wherever possible by removing the personally identifiable information in order to decrease the risk of unintended disclosure.	MUST
R.P.7	New	Yes	Data granularity MUST be reduced wherever possible, e.g. disseminate a location-related information (i.e. area) and not the exact address.	MUST
R.P.8	New	Yes	Data MUST be stored in encrypted form.	MUST
R.P.9	New	Yes	Repeated querying for specific data by applications, services, or users that are not intent to act in this manner SHALL be blocked.	SHALL
R.P.10	New	Yes	Wherever possible, information over groups of attributes or groups of individuals SHALL be aggregated (e.g. 'the majority of people that visited the examined area in this time interval were young students' this is sufficient information for an advertising application of a nearby shop, without requiring to process raw data from the personal IoT devices).	SHALL
R.P.11	New	No	The data principal SHALL be sufficiently informed regarding which data are collected, processed, and disseminated, and for what purposes	SHALL
R.P.12	New	Yes	During all communication and processing phases logging MUST be performed to enable the examination that the system is operating as promised	MUST
R.P.13	New	Yes	The user SHALL be able to control the privacy mechanisms (i.e. redemption period, data granularity and dissemination, and anonymization technique)	SHALL

Furthermore, several generic security and privacy requirements for specific SEMIoTICS components and core technologies are described in the use cases 1-10. The following table summarizes these requirements, which are also mentioned in the Appendix A.

TABLE 6: GENERIC SECURITY AND PRIVACY REQUIREMENTS FOR THE SEMIOTICS USE CASES

Req-ID	Former Req-ID	Functional	Description	Req. level
R.GSP.1	R4.4	Yes	The Intrusion Detection System (IDS) MUST capture and process suspicious traffic.	MUST
R.GSP.2	R4.5	No	Accredited and certified Computer Emergency Report Team (CERT) MAY get informed about an occurring cyber incident (e.g. DoS).	MAY
R.GSP.3	R8.4	Yes	IoT gateway SHALL be able to estimate abnormal detection based on (un)-supervised model.	SHALL
R.GSP.4	R9.1	No	Platforms, e.g. cloud platform and sensor, SHALL be trusted.	SHALL
R.GSP.5	R9.2	No	Sensors SHALL to be identifiable, i.e. they either need a TPM	SHALL

			module/smartcard, or they need a user interface.	
R.GSP.6	R9.3	Yes	Sensors MUST to be able to encrypt the data they generate, i.e. their CPU and memory need to be sufficient to perform these cryptographic operations.	MUST
R.GSP.7	R9.4	Yes	The cloud platform SHALL to be able to monitor the execution of an app, in particular its interactions with other apps, the network interface, and APIs.	SHALL
R.GSP.8	R2.18	Yes	The SARA system MUST periodically log (Hub monitored) Patient bio-medical data to the ACS.	MUST
R.GSP.9	R2.24	No	The SARA system SHALL provide robust mechanisms to protect Patient-related data - i.e. to: · Authenticate the sources of Patient data – in particular: to avoid the case that data monitored from one Patient is incorrectly ascribed to another. · Prevent unauthorised access to or manipulation of stored Patient data. · Secure the transmission of Patient data. Such data includes (but is not limited to): · System configuration data for the Patient (UC 1) · Daily activity rules for the Patient (UC 2) · Monitored daily activity data for the Patient (UC 2) · Audio-visual streams transmitted during telepresence (UC 6 & 7)	SHALL
R.GSP.10	R2.28	No	The SARA system MUST fully comply with all relevant Italian laws governing the privacy, security and storage of sensitive Patient health-related data.	MUST

In the Deliverable D4.12 (to be released on month 28) on security and privacy we will revisit the security- and privacy-related requirements and provide a fully comprehensive, brief overview if and how they were addressed and where the detailed information on the requirement verification can be found (e.g. if not in D4.12 itself then in which other deliverable); if the implementation is still ongoing at the date of release of D4.12 we will point to where the information will be recorded in future deliverables

3 DOMAIN SPECIFIC REQUIREMENTS

3.1 UC1: Wind Energy

The first use case of SEMIoTICS deals with the management and administration of a Wind Park. Normally, the Wind Turbine Controller in a Wind Park control network is an embedded or highly integrated operating system, which requires rigorously development, testing, pre-qualification and certification prior to the actual deployment in the field.

Monitoring and preventive maintenance are crucial in order to provide continuous power production and avoid malfunctions. In this use-case, the Operations and Management (O&M) team would need to record the development of the inclination of steel towers on a routine basis. This information is key to determine the level of fatigue, deformation and can be used for forecasting remaining lifetime of the steel tower. The next figure depicts the stakeholders and the interacting system components.

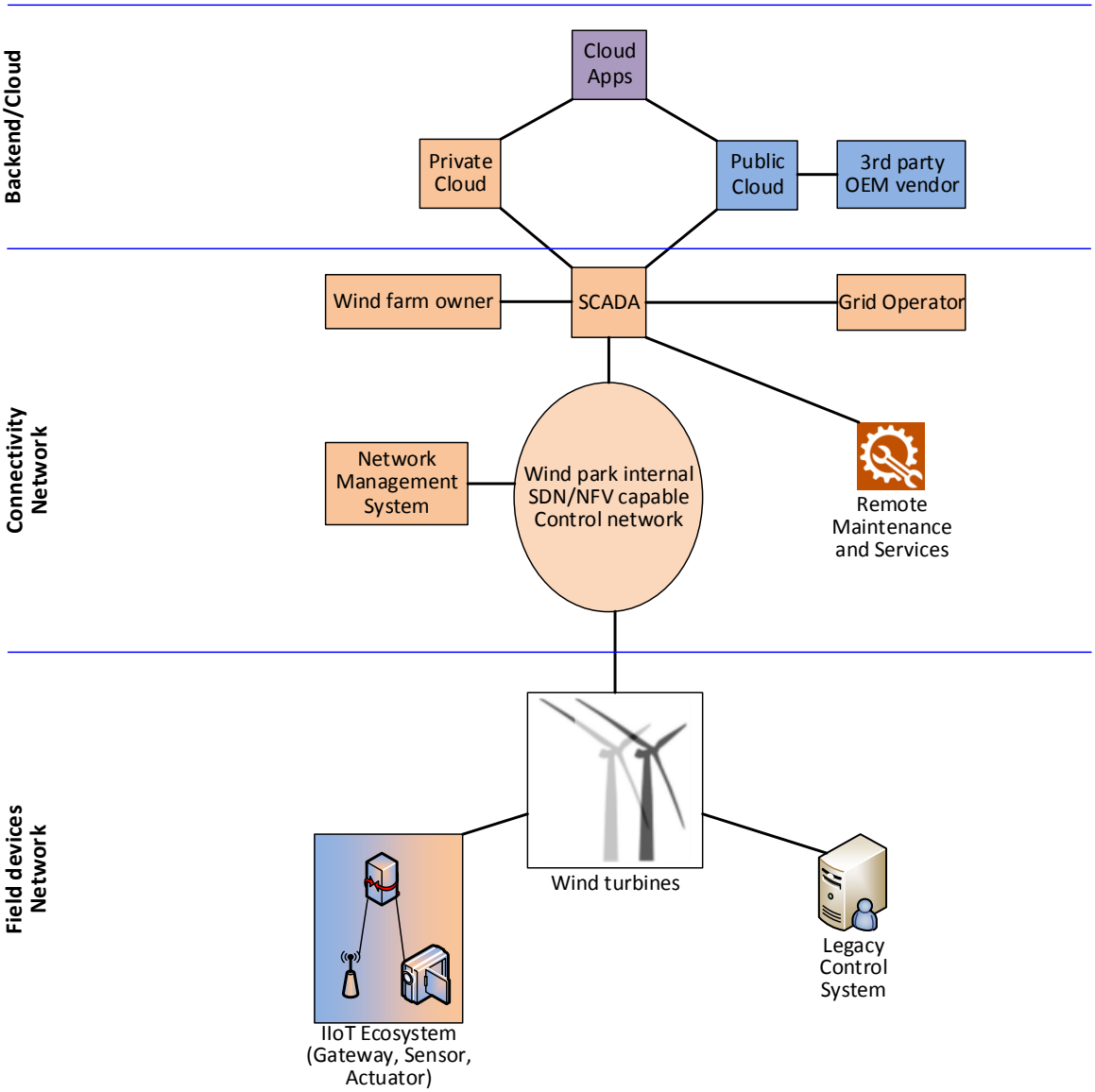


FIGURE 2: STAKEHOLDERS AT EACH LAYER IN WIND ENERGY USE CASE

This scenario examines the reliable calculation of the absolute inclination angle locally, inside the turbine, preventing limit values from being exceeded on a routine basis, while reducing the large amount of data that is sent back to the control centre. Another issue that is tackled includes the smart actuation by sensing a continuous stream of unstructured audio or video data. When the turbine rotor aligns to the wind direction

in order to maximize the energy production, then IIoT sensors can detect grease leakage or unintended noise compared to a known baseline. This sensing/detection of the unstructured data and acting locally to reduce damages to the turbine's components, in the event of a failure, is of key importance. The following table classifies the requirements of the Wind Energy prototype.

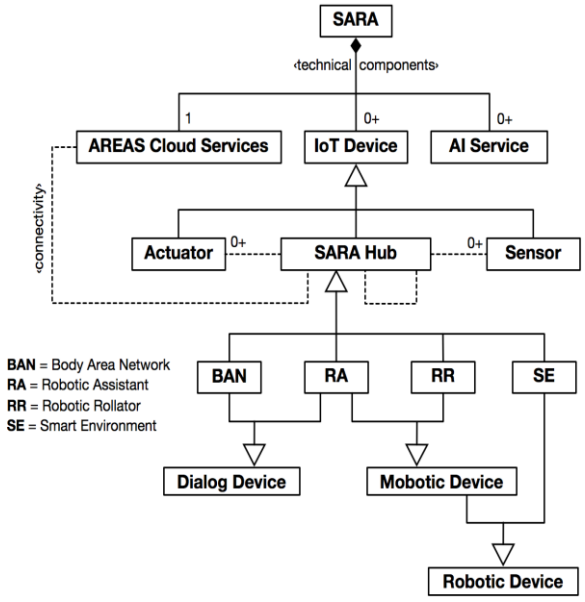
TABLE 7: UC1 REQUIREMENTS TABLE

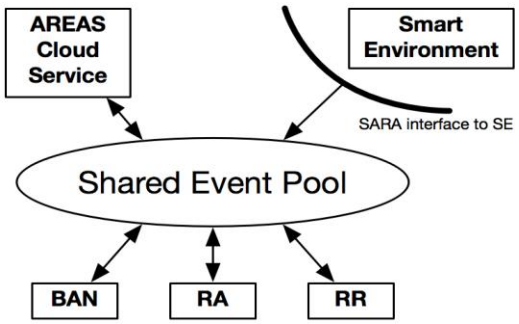
Req-ID	Former Req-ID	Functional	Description	Req. level
R.UC1.1	UC1 R1.1	Yes	Automatic establishment of networking setup MUST be performed to establish End-to-end connectivity between different stakeholders	MUST
R.UC1.2	UC1 R1.2	Yes	Automatic establishment of computing environment MUST be performed in IIoT Gateway for the minimum operation of the IIoT devices through 5G network controller based on SDN/NFV	MUST
R.UC1.3	UC1 R1.3	Yes	There MUST be enabled the definition of network QoS on application-level and automated translation into SDN controller configurations.	MUST
R.UC1.4	UC1 R1.4	Yes	Network resource isolation MUST be performed for guaranteed Service properties – i.e. reliability, delay and bandwidth constraints.	MUST
R.UC1.5	UC1 R1.5	Yes	Fail-over and highly available network management SHALL be performed in the face of either controller or data-plane failures.	SHALL
R.UC1.6	UC1 R1.6	Yes	Decisions made by unreliable, i.e. faulty or malicious SDN controllers, SHALL be identified and excluded.	SHALL
R.UC1.7	UC1 R1.7	Yes	The operation of the SDN control SHALL be scalable to cater for a massive IoT device integration and large-scale request handling in the SDN controller(s) using a (near-) optimal IoT client – SDN controller assignment procedure.	SHALL
R.UC1.8	UC1 R1.8	Yes	Semantic and robust bootstrapping/registration of IIoT sensors and actuators with IIoT gateway MUST be supported.	MUST
R.UC1.9	UC1 R1.9	Yes	Semantic interaction between use-case specific application on IIoT Gateway and legacy turbine control system MUST be supported.	MUST
R.UC1.10	UC1 R1.10	Yes	Local analytical capability of IIoT Gateway to run machine learning algorithms (e.g. specific to 2 specific sub-use cases)	MUST
R.UC1.11	UC1 R1.11	No	Device composition and application creation SHALL be supported through template approach.	SHALL
R.UC1.12	UC1 R1.12	No	Standardized semantic models for semantic-based engineering and IIoT applications MUST be utilized.	MUST
R.UC1.13	New	Yes	Middleware functionality MUST be supported on IIoT gateway, to deal with termination of IIoT sensors, signal processing and termination of interfaces to legacy systems to provide prioritization and QoS for IIoT applications.	MUST

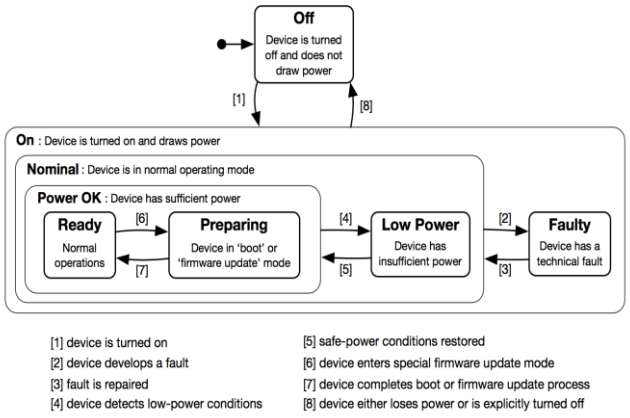
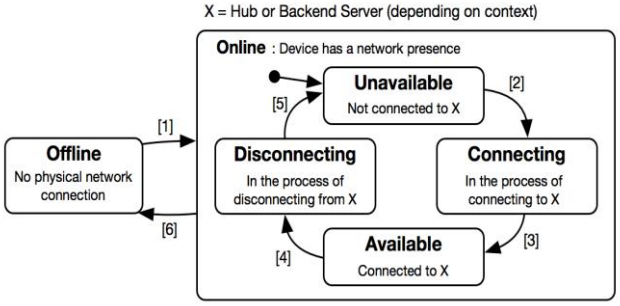
3.2 UC2: SARA Healthcare

The derived specific requirements of the SARA healthcare UC, from the ones specified in deliverable D2.2, are listed below:

TABLE 8: UC2 REQUIREMENTS TABLE

Req-ID	Former Req-ID	Functional	Description	Req. level
R.UC2.1	Derived from R2.13	Yes	The SEMIoTICS platform SHOULD support time- and safety-critical requirements by allowing SARA application logic to be deployed on resource-constrained edge gateways (e.g. smartphones, vehicles, mobile robots). SEMIoTICS platform functionalities SHOULD be locally available even in case of failure of communication with the SEMIoTICS cloud nodes.	SHOULD
R.UC2.2	Derived from R2.13	Yes	The SEMIoTICS platform SHOULD support the SARA solution to manage the trade-off between different requirements (e.g. reliability, power consumption, latency, fault-tolerance) by allowing both SARA application logic and platform features to be distributed over a cluster of gateways (SARA Hubs).	SHOULD
R.UC2.3	Derived from R2.5, R2.13	Yes	The SEMIoTICS platform SHOULD guarantee proper connectivity between the various components of the SARA distributed application. The SARA solution is a distributed application not only because it uses different cloud services (e.g. AREAS Cloud services, AI services) from different remote computational nodes, but also because the SARA application logic itself is distributed across various edge nodes (SARA Hubs). The following diagram shows the components of the SARA system: 	SHOULD

R.UC2.4	Derived from R2.14	Yes	The SEMIoTICS platform SHOULD provide services to synchronize and coordinate the activities of the various components of the SARA application. The SARA components synchronize and coordinate their activities by relying on the “shared event pools” paradigm. Through a “shared event pool”: • All devices contribute events to the pool – e.g. "Patient has requested escort to kitchen", "Patient is in distress". • The AREAS Cloud Service (ACS) manages consistency of data in the pool. • The pool ensures that all devices share the same operational context. • The pool allows multiple devices to respond independently to the same events (redundancy). • The design & development of the high-level system functions is simplified: abstracts away from low-level device implementation & communication details. 	SHOULD
R.UC2.5	Derived from R2.23	Yes	The SEMIoTICS platform should allow the SARA solution to discover the IoT devices that are registered in the system. IoT devices deployed by the SARA solution are expected to register themselves into the system using various standard protocols (e.g. LwM2M, MQTT, Bluetooth LE, ZigBee, etc.).	SHOULD
R.UC2.6	Derived from R2.22	Yes	The SEMIoTICS platform SHOULD allow the SARA solution to retrieve the resources exposed by registered devices via their object model (i.e. a data structure wherein each element represents a resource, or a group of resources, belonging to a device). The SEMIoTICS platform SHOULD support at least the OMA LWM2M object model.	SHOULD

R.UC2.7	Derived from R2.13	Yes	The SEMIoTICS platform SHOULD notify periodically the SARA solution about the state of the resources hosted by registered IoT field devices. The following diagrams present a draft state diagram for the IoT field devices managed by SARA:  [1] device is turned on [2] device develops a fault [3] fault is repaired [4] device detects low-power conditions [5] safe-power conditions restored [6] device enters special firmware update mode [7] device completes boot or firmware update process [8] device either loses power or is explicitly turned off	SHOULD
R.UC2.8	Derived from R2.22	Yes	The SEMIoTICS connectivity SHOULD keep track of the field device connectivity state (e.g. to detect anomalies, but also required for higher-level (cognitive) control algorithms). The following figure shows a draft of the connectivity states and transitions expected by the SARA solution:  [1] physical network connection created [2] device initiates connection to X [3] device establishes connection with X [4] device initiates disconnection from X [5] disconnection from X complete [6] physical network connection lost	SHOULD
R.UC2.9	Derived from R2.17	Yes	The SEMIoTICS platform SHOULD allow the SARA solution to persist and, subsequently, retrieve the notifications received by IoT field devices. The SARA components expect the values to be stored as key-value pairs within collections belonging to data stores.	SHOULD

R.UC2.10	Derived from R2.2	Yes	The SEMIoTICS platform SHOULD allow the SARA components (e.g. SARA Hubs) to query and aggregate (e.g. to average) the values of a resource (e.g. current measured temperature) hosted by a group of field devices. The SARA solution defines a group of devices by specifying filtering criteria over the set of registered devices. These filtering criteria may concern the location of the device, the Quality of Service (QoS) offered (e.g. its precision) and their ownership. The SARA components MAY submit three classes of queries: one-time, periodic, and conditional. One-time queries are evaluated once. Their evaluation starts at the time of the submission and their results are retrieved asynchronously by the application. With periodic query, a SARA component requests periodic evaluation and reporting. With conditional queries, the application is notified whenever a Boolean condition, defined over the value computed by the query, changes its truth value.	SHOULD
R.UC2.11	Derived from R2.30	Yes	The SEMIoTICS platform SHOULD allow a SARA component to request a group of devices to take/initiate an action (e.g. turn on/off a light bulb).	SHOULD
R.UC2.12	Derived from R2.5 R2.6 R2.28	Yes	The SEMIoTICS platform SHOULD allow SARA components to delegate to the platform the computation of complex functions over the data received by field devices. These computations may result either in the generation of higher-level observation events (e.g. significant Patient events abstracted form sensor data) towards the ACS or in sensors configuration parameters (including actuators command). The SARA components MAY specify computations either as Dataflow or as Finite State Machine.	SHOULD
R.UC2.13	Derived from R2.28	Yes	The SEMIoTICS platform MAY allow the SARA solution to synchronize the clocks of the mobile robotic devices hosting the components of the solution. This service, hence, would enable the SARA solution to properly order the events generated by its components (e.g. to generate a single coherent map of the environment starting with data from multiple hubs). Moreover, the SEMIoTICS platform MAY allow the SARA components to delegate the management of timers relying on the synchronized clocks.	MAY
R.UC2.14	Derived from R2.5 R2.12	Yes	The SEMIoTICS platform MAY allow the SARA solution to access information concerning the location of mobile robotic nodes and to maintain SARA specific location information (e.g. collected map data). This would allow the SARA hub or cloud to send to robotic devices high level navigation related commands (e.g. “go to location”).	MAY

R.UC2.15	Derived from R2.14 R2.15 R2.21	No	The SEMIoTICS platform SHOULD provide low latency connectivity between the SARA hubs and cloud services (i.e. AREAS cloud services and AI services) to allow offloading of near real-time computation intensive tasks to the cloud. Examples include: • the robotic assistant (RA) employing AI services to analyse Patient's speech (audio) and body language (video) to identify significant events – e.g. "Patient requests an escort", "Patient asks where his glasses are • the robotic rollator (RR) exploiting AI Services to analyse Patient's gait and posture to identify significant events – e.g. "Patient has fallen". • mobile robotic Devices (RA/RR) exploiting cloud resources for simultaneous localization and mapping (SLAM) Therefore, SARA hubs need to send with minimal delay: • raw range data (e.g. from Lidar sensors) to identify proximal objects/objects, • real-time audio stream for speech analysis, • and real-time raw video stream (object/people recognition, gesture recognition, posture analysis).	SHOULD
R.UC2.16	Derived from R2.2 R2.22 R2.23	No	The SEMIoTICS platform SHOULD support secure bootstrap, configuration and diagnostic of SARA hubs and devices.	SHOULD
R.UC2.17	Derived from R2.15 R2.21	No	The SEMIoTICS connectivity SHOULD support real time exchange of raw sensor data among sensors/actuators and SARA Hubs.	SHOULD

3.3 UC3: IHES Artificial Intelligent Sensor for Event Detection

The Artificial Intelligent Sensor for Event Detection use case is defined within SEMIoTICS has an horizontal use case enabling computation "on the edge" (i.e. on low power 32 bits MCU units), by deploying on these devices AI algorithms, statistical & analytical models, ad-hoc customizable SW infrastructures that allows to self-learn and self-adapt to a generic time variant signal coming from a sensor stream. The Figure 3 depicts how this UC will be deployed on top of the SEMIoTICS infrastructure at Field Devices Level, where all complete set of functionalities for the demonstrator could be efficiently and conveniently mapped.

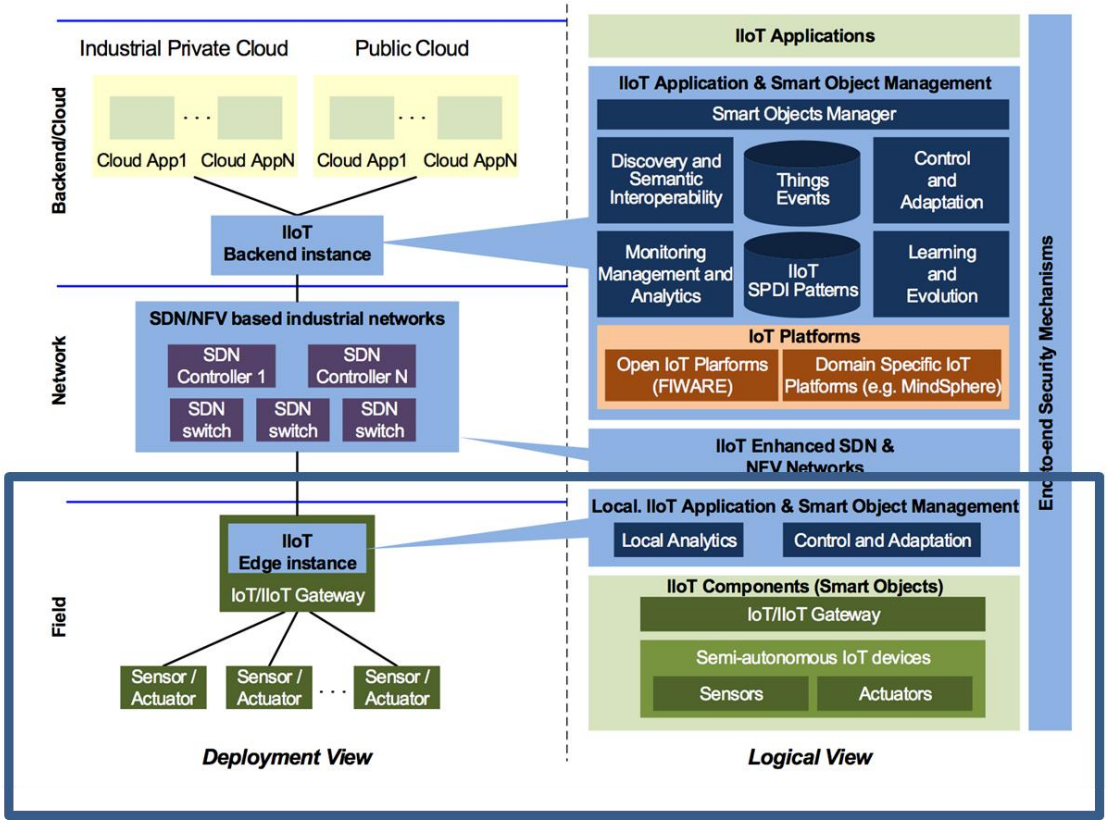


FIGURE 3: HORIZONTAL UC3 DEPLOYMENT ON SEMIOTICS ARCHITECTURE

A more detailed overview of the main functionalities of the UC are shown on Figure 4 and the derived specific requirements are detailed in Table 9.

The Figure 4 introduces the UC3 as a scenario where an IoT distributed system composed by several intelligent sensing units are coordinating with an intelligent gateway through an IP based 1 to many M2M communication protocol. The scenario envisages the capability of the system to self-learn (i.e. online training) a predictive model from a generic time variant signal acquired from a sensor tightly coupled with the MCU. To enable this intelligent behaviour of self-learning and self-adaptability several algorithms from analytical model theory, AI, Statistical model, etc. will be implemented partitioning them between the MCU units and the gateway. This system will be able to detect relevant events and anomalies from the learned model without the need on any kind of supervision.

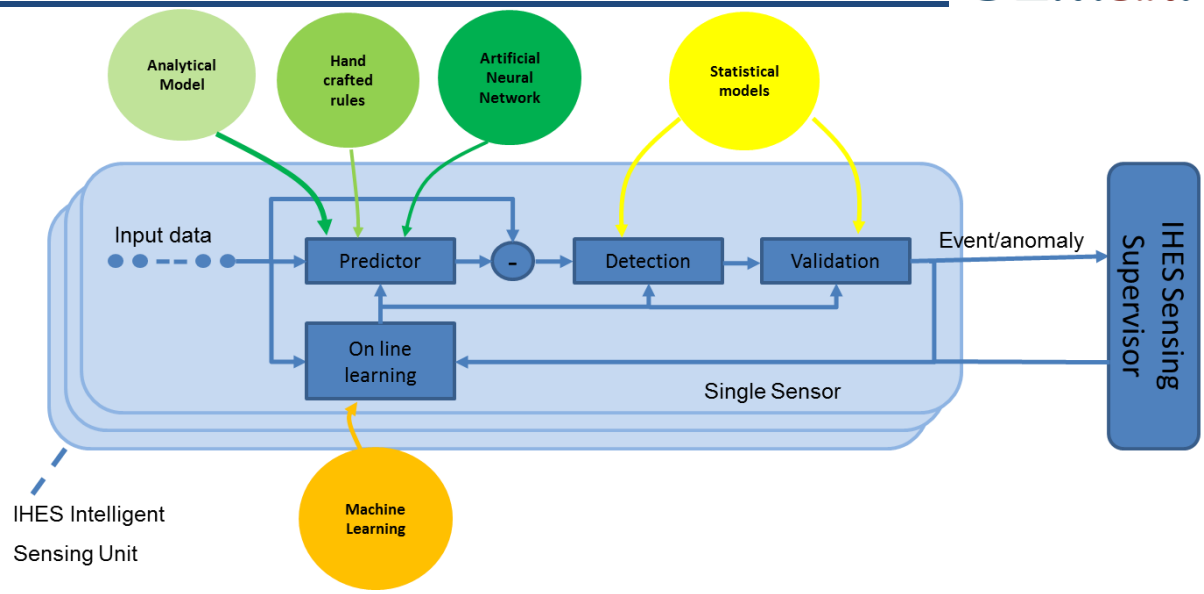


FIGURE 4: UC3 FUNCTIONAL DIAGRAM OVERVIEW

The requirements specified in deliverable D2.2 and the general description of section 2.4 are further detailed into use case specific requirements in Table 7:

TABLE 9: UC3 REQUIREMENTS TABLE

Req-ID	Former Req-ID	Functional	Description	Req. level
R.UC3.1	Derived from R3.1	Yes	IoT Sensing unit shall be able to embed environmental (e.g. temperature, pressure, humidity, light) and inertial sensors (accelerometer, gyroscope).	SHOULD
R.UC3.2	Derived from R3.2	Yes	IIoT Sensing unit shall be able to interface to the IIoT Sensing gateway in order to coordinate with it. A standard IP based (i.e. TCP transport) 1 to many M2M communication protocol must be adopted to properly handle node communication with components in the gateway.	MUST
RUC3.3	Derived from R3.3	Yes	IIoT Sensing unit shall be able to learn a model from observed data in an unsupervised manner. In particular IoT Sensing unit shall be equipped with a low power (tens/hundreds of mW range) 32 bits MCU to support unsupervised learning and unsupervised statistical processing.	MUST
R.UC3.4	Derived from R3.4	Yes	IIoT Sensing unit shall be able to detect relevant changes from the learned model and report them to IIoT Sensing gateway.	MUST
R.UC3.5	Derived from R3.5	No	IIoT Sensing unit shall be able to adapt to a new model if IIoT sensing gateway requires this.	MUST
R.UC3.6	Derived from R3.6	Yes	IIoT Sensing gateway shall be able to coordinate a set of IIoT sensing units by finding any correlation btw them according to observed data, models	MUST
R.UC3.7	Derived from R3.7	Yes	IIoT Sensing gateway shall be able aggregate relevant events (i.e. changes) coming from whichever of connected IIoT sensing units deciding if they are global or local changes	MUST

R.UC3.8	Derived from R3.8	Yes	IIoT Sensing gateway may have the capability to exchange relevant information (i.e. events) between itself, the cloud and the sensing units with some connectivity capabilities	SHOULD
R.UC3.9	Derived from R3.9	Yes	IIoT Sensing web GUI may be able to display correlations between connected IIoT Sensing units and the status related to each IIoT sensing unit.	MUST
R.UC3.10	Derived from R3.10	Yes	IIoT Sensing web GUI may be able to display logging about relevant events detected by connected IIoT Sensing units reporting info about unit ID, type of data and type of event detected.	MUST
R.UC3.11	Derived from R3.5	Yes	IoT Sensing unit shall be able to run Artificial neural networks on the MCU in real time at the sensor data rate of choice.	MUST
R.UC3.12	Derived from R3.5	Yes	IoT Sensing unit shall be able to run lightweight statistical model analysis algorithms on the MCU not in real time at the sensor data rate of choice.	SHOULD
R.UC3.13	New	Yes	MCU into IoT Sensing unit should be associated with a high-level tool for automatic generation of optimized code to support pre trained neural networks.	MAY
R.UC3.14	Derived from R3.5	Yes	MCU IoT Sensing unit shall be able to run neural network online training at the sensor data rate of choice.	MUST
R.UC3.15	Derived from R3.2	Yes	IoT Sensing gateway shall support 1 to many standard IP based (i.e. TCP transport) M2M communication protocol to interface a number N of connecting Sensing units (e.g. broadcast type).	MUST
R.UC3.16	New	Yes	IoT Sensing gateway shall be equipped with a low power embedded CPU (100s of mW to W).	MUST
R.UC3.17	Derived from R3.2	Yes	IoT Sensing gateway shall be able to negotiate capabilities, notify, start and shutdown any Sensing unit at any point in time.	SHOULD
R.UC3.18	New	Yes	IoT Sensing gateway shall be capable to run Linux (e.g. Ubuntu OS) and standard graphics and browser libraries.	MUST
R.UC3.19	Derived from R3.8	Yes	IoT Sensing gateway should be able to support http and standard protocols for cloud interfacing.	SHOULD
R.UC3.20	Derived from R3.2	Yes	The specific M2M protocol adopted on UC3 is based on MQTT. A MQTT broker service will be available to dispatch messages between the coordinating Sensing gateway and its associated Sensing units.	MUST
R.UC3.21	Derived from R3.2	Yes	A use case specific serialized message protocol is required to coordinate the gateway and its associated units and exchange data / events / anomalies between them. JSON will be the preferred serialization format adopted.	SHOULD
R.UC3.22	Derived from R3.2	Yes	Each connected IHES sensing unit should send to the gateway a keep alive signal on a specified period (e.g. few seconds) to notify the gateway it is correctly working. The sensing gateway should detect by this mean any non-working sensing unit and reconfigure the system accordingly.	SHOULD
R.UC3.23	Derived from R3.2	Yes	Sensing units and sensing gateway should share a common clock (i.e. global reference time), precise up to milliseconds, to properly classify events and data acquired during the processing. This global reference time will be negotiated when a sensing	SHOULD

			unit node will join a given gateway. Internally the system will work scheduling activities according to this global reference time.	
R.UC3.24	New	Yes	Sensing units may be equipped with dedicated FW to detect relevant sensors malfunctioning and report that to the gateway	MAY

4 CONCLUSIONS

This chapter performs an initial analysis on the project's implementation framework, and specifically, the technologies which are and will be used during the design, implementation, integration and deployment of the SEMIoTICS prototype. This first evaluation of available technologies and platforms derives from the requirements and architecture design presented in the previous chapters and will be the basis for the technology scouting and final definition of the implementation framework, performed in upcoming work WP3, WP4, and WP5. The implementation framework includes all the tools which are required to design, develop, integrate and test the SEMIoTICS system and will be finalized in the next three WPs:

- **WP3:** Smart objects and networks
- **WP4:** Pattern-driven smart behaviour of IIoT/IoT with end-to-end security and privacy
- **WP5:** System integration and evaluation monitoring mechanisms

4.1 Logical link with WP3 beneficiaries

WP3 deals with two main objectives. First, it will envisage and develop mechanisms to employ and to extend the benefits of SDN/NVF technologies for IIoT/IoT systems. The SDN mechanisms will permit the orchestration of the network for different QoS requirements demanded by the IoT applications (e.g. in terms of bandwidth or latency). The NFV approach will lead to a more efficient use of the network. NVF will also remove complexity of local and access networks by virtualizing network infrastructure and by centralizing at the backend servers the management of network functions. The requirements described in D2.3 that affect more the WP3 tasks dealing with SDN and NFV are the ones that focus on the networking requirements. Namely, these include:

- General Platform/technological and Infrastructure requirements.
- Functional and non-functional requirements at the Network level.
- Domain specific requirements.

Moreover, the WP3 tasks dealing with SDN and NFV are affected by the other requirements in D2.3 in terms of interaction between the SEMIoTICS architecture layers and functionalities. These other tasks also define several SDPI patterns that affect the network layer functionality.

The second main objective of WP3 is to leverage semantic information to obtain a more efficient interface between field devices and the IoT applications using them. This semantic information includes a vocabulary to describe the field devices in terms of properties, actions or events. Also, it includes the functional description of their interfaces, which are machine-interpretable and permit IoT applications, such as field automation systems, to interact with field devices. Moreover, the efficient end-to-end semantic interface demands that the network services APIs are semantically described as well. This semantic-driven interface between field devices and IoT applications affects all the SEMIoTICS architecture. Thereby, the requirements stemming from the "2.1 General Platform/technological/infrastructure requirements" section of D2.3 are the ones that will affect more the design of this second objective of WP3. Furthermore, the "domain specific requirements" are important as they define requirements stemming from the specific SEMIoTICS use cases.

4.2 Logical link with WP4 beneficiaries

WP4 will tackle some of the main objectives that are examined by the project. This includes, among others, the development of:

- Patterns for orchestration of smart objects with guaranteed SPDI properties
- Dynamic, self-adaptable, and predictive monitoring
- Multi-layered embedded intelligence
- End-to-end protection
- Semantic interoperability mechanisms

The general functional and non-functional requirements affect almost all WP4 perspectives in the various layers and building-blocks of SEMIoTICS. The specific requirements for each objective are covered in at least one of the described use cases, where: SPDI patterns administrate a smart building infrastructure in UC4, dynamic and self-adaptable mechanisms monitor the energy consumption of modern smart cities in UC7, edge analytics and embedded intelligence enable smart sensing in UC8, security and privacy techniques enhance end-to-end protection of smart wearable devices that communicate information to the cloud in UC9, and semantic interoperability technologies for industrial automation tackle the integration of components for several vendors in UC10.

4.3 Logical link with WP5 beneficiaries

WP5 has four key objectives:

- Define a framework of criteria for evaluating the SEMIoTICS approach and platform from a technical, business, and usability perspective.
- Define and execute proof-of-concept experiments (in-lab) and trials (real-world) to qualitatively and quantitatively assess the performance of and gains from the adoption of the developed platform.
- Evaluate the proposed framework in terms of innovation, user experience and acceptance potential.
- Validate the benefits provided (to all involved stakeholders) by the developed platform and IIoT/IoT applications under a well-defined set of diverse use-cases, based on the identified target measures and assessment criteria.

The requirements by the use cases triggered in this deliverable will allow to experiment, evaluate and validate in depth the SEMIoTICS platform in the context of WP5 since they deal with all the objective of the project. In particular, 29 of the one hundred use case requirements deals with SPDI patterns, 36 with SDN and NFV, 48 deals with IoT Platform, 29 have to do with monitoring and adaptation, 22 concern machine learning, 27 challenge security and privacy and 29 touches semantic interoperability.

5 APPENDIX A

The following table summarizes the initial requirements set for all the use cases, as they had been detailed mainly in the deliverable D2.2. The initial requirement set includes both the SEMIoTICS main use cases and the Non-SEMIoTICS use cases according to the definition provided in D2.2.

TABLE 10: THE SPECIFIC REQUIREMENTS FOR THE USE CASES 1-10

UC	Requirements				Section 2								
			Functional (F)	Non-Functional (NF)	General Platform / Technological / Infrastructure	Backend / Cloud		Network Level		Field/ Devices		IoT Security/ Privacy	
	Req-ID	Description	F	NF		F	NF	F	NF	F	NF	F	NF
UC1	R1.1	Automatic establishment of networking setup MUST be performed to establish end-to-end connectivity between different stakeholders.	●					●					
	R1.2	Automatic establishment of computing environment SHALL be performed in IIoT Gateway for the minimum operation of the IIoT devices through 5G network controller based on SDN/NFV.	●					●					
	R1.3	There MUST be enabled the definition of network QoS on application-level and automated translation into SDN controller configurations.	●					●					
	R1.4	Network resource isolation SHALL be performed for guaranteed Service properties – i.e. reliability, delay and bandwidth constraints.	●					●					
	R1.5	Fail-over and highly available network management MUST be performed in the face of either controller or data-plane failures.	●					●					
	R1.6	Decisions made by unreliable, i.e. faulty or malicious SDN controllers, MUST be identified and excluded.	●					●					
	R1.7	The operation of the SDN control SHALL be scalable to cater for a massive IoT device integration and large-scale request handling in the SDN controller(s) using a (near-) optimal IoT client – SDN controller assignment procedure.	●					●					
	R1.8	Semantic and robust bootstrapping/registration of IIoT sensors and actuators with IIoT gateway SHALL be supported.	●							●			
	R1.9	Semantic interaction between IIoT Gateway and legacy turbine control system SHALL be supported.	●							●			
	R1.10	Local analytical capability of IIoT Gateway to run machine learning algorithms (e.g. specific to 2 specific sub-use cases)	●							●			
	R1.11	Device composition and application creation SHALL be supported through template approach.		●							●		
	R1.12	Standardized semantic models for semantic-based engineering and IIoT applications SHALL be utilized.		●			●		●		●		
UC2	R2.1	The SARA system MUST provide a UI for the GP to define system configurations for each Patient.	●										

	R2.2	The SARA system MUST provide a UI for the Technician to access system configuration data for Patients.	●												
	R2.3	The SARA system MAY provide a UI (e.g. on a remote client) to allow the Technician to configure the system for a Patient.	●												
	R2.4	The SARA system must provide a UI for the GP to define 'daily activity rules' for each Patient.	●												
	R2.5	The SARA system MUST (via the SARA Hubs) continuously monitor the Patient's daily activities.	●												
	R2.6	The SARA system MUST validate monitored Patient activities against the prescribed 'daily activity rules'.	●												
	R2.7	The SARA system MUST provide (via Dialog devices) timely activity suggestions to the Patient – e.g. for scheduled events (taking medicine, programmed exercises) and in case of 'violations' of the 'daily activity rules'.	●												
	R2.8	The RA MUST assist the Patient in performing a variety of cognitive rehabilitation exercises.	●												
	R2.9	The SARA system MUST recognize (via Dialog devices) and respond appropriately to Patient requests to perform cognitive exercises.	●												
	R2.10	The SARA Hubs MUST be able to detect, recognize, identify and determine the positions of significant objects (including people) in the environment.	●												
	R2.11	The SARA system MUST keep track of the changing positions of significant objects (including people) in the environment.	●												
	R2.12	The SARA system MUST recognize (via Dialog devices) and respond appropriately to Patient requests to: i) be informed of locations of objects/people, ii) Receive physical support from the RR, and iii) be escorted to a location by the RA.	●												
	R2.13	The SARA system MUST (via SARA Hub devices) continuously monitor the Patient to detect, and raise an alert in case of, incidents (i.e. critical medical events such as falls, heart attacks, etc.).	●												
	R2.14	The SARA system MUST reliably and rapidly notify assigned Caregivers of Patient incidents.	●												
	R2.15	The SARA system MUST (in conjunction with the RA) provide on-demand telepresence (bi-directional, real-time, high-quality, audio-video streaming) between the RA's tablet device (Patient side) and either the Caregiver's mobile phone (UC 6) or GP's desktop computer (UC 7).	●												
	R2.16	The SARA system MUST (for reasons of privacy) notify the Patient whenever a telepresence session has started/ended, and also inform the Patient of the identity of the remote operator.	●												

	R2.17	The SARA system MUST provide a UI for the GP to access historical logs of Patient bio-medical data from the ACS.	●												
	R2.18	The SARA system MUST periodically log (Hub monitored) Patient bio-medical data to the ACS.	●											●	
	R2.19	The SARA system MUST recognize (via Dialog devices) and respond appropriately to Patient requests for physical exercises.	●												
	R2.20	The SARA system MUST assist the Patient in performing a variety of physical rehabilitation exercises.	●												
	R2.21	The SARA system MUST provide (via the SARA Hubs) the GP remote access to real-time (Hub monitored) Patient biomedical data.	●												
	R2.22	The SARA system SHOULD support and track the dynamic addition/removal of system components (e.g. health monitors temporary added to the BAN).	●												
	R2.23	The SARA system SHOULD facilitate the integration of legacy and new IoT Devices (Hubs, Sensors & Actuators) - e.g. through standardized low-level API conformance.	●												
	R2.24	The SARA system should provide robust mechanisms to protect Patient-related data - i.e. to: · Authenticate the sources of Patient data – in particular: to avoid the case that data monitored from one Patient is incorrectly ascribed to another. · Prevent unauthorized access to or manipulation of stored Patient data. · Secure the transmission of Patient data. Such data includes (but is not limited to): · System configuration data for the Patient (UC 1) · Daily activity rules for the Patient (UC 2) · Monitored daily activity data for the Patient (UC 2) · Audio-visual streams transmitted during telepresence (UC 6 & 7)		●										●	
	R2.25	The communication of health-related Patient data between SARA components SHOULD employ/conform to HL7 protocols.		●											
	R2.26	The SARA system components and communications technologies MUST NOT interfere with, and MUST NOT be adversely affected by, medical equipment.		●											
	R2.27	The SARA system MUST perform with consistent quality (as assured through version control of source code and strict, systematic and high-coverage regression testing) in all target operating environments.		●											
	R2.28	The SARA system MUST fully comply with all relevant Italian laws governing the privacy, security and storage of sensitive Patient health-related data.		●											●

	R2.29	The SARA system MUST ensure that all health-critical communications (in particular: incident alerts) between components are prioritized in terms of receiving network resources and ensuring reliable transmission to destination.		●											
	R2.30	The SARA system MUST reliably and rapidly respond appropriately to health-critical events (in particular incident alerts).		●											
	R2.31	The SARA robotic devices SHOULD NOT (as far as this is preventable) cause any harm or damage to individuals and objects while navigating around the home.		●											
	R2.32	The SARA robotic devices MUST include emergency failsafe 'stop' mechanisms allowing them to be shut down in case of impending or actual risk to persons or objects.		●											
UC3	R3.1	IIoT Sensing unit shall be able to interface to the physical world	●								●				
	R3.2	IIoT Sensing unit shall be able to interface to the IIoT Sensing gateway in order to coordinate with it.	●								●				
	R3.3	IIoT Sensing unit shall be able to learn a model from observed data in an unsupervised manner		●								●			
	R3.4	IIoT Sensing unit shall be able to detect relevant changes from the learned model and report them to IIoT Sensing gateway	●								●				
	R3.5	IIoT Sensing unit shall be able to adapt to a new model if IIoT sensing gateway requires this.		●								●			
	R3.6	IIoT Sensing gateway shall be able to coordinate a set of IIoT sensing units by finding any correlation btw them according to observed data, models	●								●				
	R3.7	IIoT Sensing gateway shall be able aggregate relevant events (i.e. changes) coming from whichever of connected IIoT sensing units deciding if they are global or local changes	●								●				
	R3.8	IIoT Sensing gateway may have the capability to exchange relevant information (i.e. events) between itself, the cloud and the sensing units with some connectivity capabilities	●								●				
	R3.9	IIoT Sensing web GUI may be able to display correlations between connected IIoT Sensing units and the status related to each IIoT sensing unit.	●								●				
	R3.10	IIoT Sensing web GUI may be able to display logging about relevant events detected by connected IIoT Sensing units reporting info about unit ID, type of data and type of event detected.	●								●				
UC4	R4.1	Connectivity of the SDN controller with the various underlying components (e.g. the SDN switches and the IoT gateways and devices) in order to support the automated configuration process		●						●					
	R4.2	Connectivity of the SDN controller with the remote management service in order to support the remote configuration process		●						●					

	R4.3	The SDN controllers must be always available to serve incoming requests		●				●						
	R4.4	Intrusion Detection System (IDS) that captures and processes suspicious traffic	●				●				●			
	R4.5	Accredited and certified Computer Emergency Report Team (CERT) MAY get informed about an occurring cyber incident (e.g. DoS)		●	●								●	
UC5	R5.1	Low latency and reliable communication between the energy controllers. This communication requires <10ms latency. This is mandatory for the iterations required by the distributed optimization carried out by the energy controllers VNFs, when they are deployed at the IoT GWs.		●				●						
	R5.2	Interoperability between the IoT Gateways and the IoT devices (sensors/actuators), due to the heterogeneous intranets communicating them.		●				●						
	R5.3	Low latency and reliable communication between the sensor measurements and the IoT gateway.		●				●						
	R5.4	Low latency and reliable communication between the energy grid operator and the energy controllers.		●				●						
	R5.5	Low latency and reliable communication between the energy controller VNFs, within the IoT gateway, and the actuators, which manage the loads, when the VNFs are deployed at the IoT GWs. Or between the IoT backend and the buildings' actuators, when the VNFs are deployed at the IoT backend.		●				●						
	R5.6	Low latency and reliable communication between the IoT GWs and the IoT backend, when the energy controller VNF is deployed at the IoT backend.		●				●						
UC6	R6.1	IoT Gateway must be able to establish connection with devices via OPC protocol												
	R6.2	Connectivity between the control devices and IoT Cloud components for automated configuration purposes												
	R6.3	System scales even when new power plant block is monitored and managed												
	R6.4	Cloud applications to manage power plant can handle control process for many independent power blocks												
	R6.5	Cloud component with advanced computing module have to generate business events for next processing. IoT platform will decide, what changes should be done.												
	R6.6	Advanced computing module needs to be developed based on SPDI patterns which will allow connect any chosen IoT platform.												
	R6.7	IoT platform should provide possibility to report list of executed actuation commands.												
UC7	R7.1	Ensure high connectivity probability (>99%) among all the communication network devices												

	R7.2	The local cloud should be able to recognize the established patterns and act in less than 10ms														
	R7.3	The energy management system has to be able to handle data from a massive collection of smart meters, i.e. several micro-grids employ the same energy management system, and be able to scale when new micro-grids are included in the network														
	R7.4	The demand response algorithms have to decrease the energy consumption of the micro-grid by 40%														
	R7.5	The adaptive monitoring should be able to identify changes in the patterns and readapt the established actions in a timely manner														
	R7.6	The energy management system should monitor and guarantee a smooth transition from grid connection mode, to islanding mode, where the micro-grid consumes its own generated energy														
	R7.7	When there are discrepancies in the load of different buildings, the energy management system should guarantee the flow of energy from the low-load buildings to the high-load buildings to achieve load balancing														
UC8	R8.1	IoT sensing unit shall be able to interface to the physical world		●										●		
	R8.2	IoT sensing unit shall be able to interface to the IoT gateway in order to coordinate with it		●										●		
	R8.3	IoT sensing unit shall be able to perform distributed feature extraction/selection from the acquired data	●									●				
	R8.4	IoT gateway shall be able to estimate abnormal detection based on (un)-supervised model	●						●				●			
	R8.5	IoT gateway shall be able aggregate relevant events (i.e. changes) coming from whichever of connected IoT sensing units deciding if they are global or local changes	●						●							
	R8.6	IoT gateway shall have the capability to exchange relevant information between itself, the cloud and the sensing units with some connectivity capabilities	●						●							
	R8.7	Cloud/shall be able to aggregate features/data coming from the edge level through IoT gateway	●						●							
	R8.8	Cloud/shall be able to perform prediction based on the aggregated data based on a supervised model	●		●											
	R8.9	Cloud/shall be able to send updated model data/parameters to the IoT gateway	●						●							
	R8.10	IoT gateway shall be able to update the monitoring model based on the information coming from the cloud	●						●							
UC9	R9.1	Platforms, e.g. cloud platform and sensor, can be trusted.														●
	R9.2	Sensors need to be identifiable, i.e. they either need a TPM module/smartcard, or they need a user interface.														●
	R9.3	Sensors need to be able to encrypt the data they generate, i.e. their CPU and memory need to be sufficient to perform these cryptographic operations.											●			

	R9.4	The cloud platform needs to be able to monitor the execution of an app, in particular its interactions with other apps, the network interface, and APIs.										●	
UC10	R10.1	Semantic schema to enable semantic description of filed devices and edge devices.											
	R10.2	Semantic schema/metadata to enable semantic description of network (SDN and NFV) infrastructure.											
	R10.3	Semantic schema to enable application semantics in Backend/Cloud.											
	R10.4	Semantic format to describe assets (e.g., W3C WoT Thing Description)											
	R10.5	Semantic mapping between various semantic models, e.g., OPC UA IM, MindSphere IoT Model, W3C WoT TD etc.											
	R10.6	Infrastructure/tools to support semantic discovery.											
	R10.7	Infrastructure/tools to support orchestration of (application, network and other resources) based on Recipes.											
	R10.7	Infrastructure/tools to support semi-automated device Plug & Play and bootstrapping.											