Self Assessment Towards Optimization of Building Energy

Deliverable D1.3

SATO Platform, SRI and IT Security Requirements

Deliverable Lead: FC.ID

Deliverable due date: 28/02/2021

Actual submission date: 07/03/2021

Version: v2.0

This project receives funding in the European Commission's Horizon 2020 Research Programme under Grant Agreement Number 957128.

Document Control Page	
Title	SATO Platform, SRI and IT Security Requirements
Editor	FCIENCIAS.ID - ASSOCIACAO PARA A INVESTIGACAO E DESENVOLVIMENTO DE CIENCIAS
Description	This deliverable will specify in detail all the requirements for the SATO platform, for SRI assessment, and to guarantee IT security up to known standards.
Contributors	Authors: Vinicius Cogo, Vasco Ferreira, André Gil, José Cecílio, Pedro Ferreira (FC.ID) Other contributors: Michael Skovsgaard (FB), Gerhard Engelbrecht (SAGOE), Per Kvols Heiselberg (AAU), Paul Kessler, João Bravo Dias, Filipe Neves Silva, Pedro Almeida and João Wang (EDP CNET), Philippe Bollinger and Gonçalo Vilela (AMES), Guilherme Carrilho da Graça and Daniel Albuquerque (FC.ID), Ricardo Gustavo Barros and Nuno Coutinho Gouveia (SONAE)
Creation date	01/02/2021
Type	Report
Language	English
Audience	☒ public ☐ confidential
Review status	☐ Draft ☒ WP leader accepted ☒ Coordinator accepted
Action requested	☐ to be revised by Partners ☐ for approval by the WP leader ☐ for approval by the Project Coordinator ☐ for acknowledgement by Partners

Table of Contents

List of Figures

List of Tables

Revision history

Version	Author(s)	Changes	Date
v1.0	Vinicius Cogo (FC.ID)	Initial Version	01/02/2021
V1.1	Vasco Ferreira (FC.ID), André Gil (FC.ID), Vinicius Cogo (FC.ID)	Review of existing initiatives in interoperable systems, description of use cases and pilots	10/02/2021
V1.3	Vinícius Cogo (FCI.ID), José Cecílio (FC.ID)	First version of the functional and non-functional requirements	15/02/2021
V1.4	All	Final draft	25/02/2021
V1.5	Pedro Ferreira	Internally reviewed draft	28/02/2021
V2.0	Pedro Ferreira, Vinícius Cogo, Vasco Ferreira, André Gil	Version integrating changes from reviewers (EDP CNET and SAGOE)	07/03/2021

DISCLAIMER

EXECUTIVE SUMMARY / ABSTRACT / SCOPE

The SATO project aims to create a new *energy self-assessment and optimisation platform* (i.e., the *SATO platform*) that integrates and monitors energy consumption of building equipment and appliances. This platform will support a self-assessment framework (SAF) and the optimisation services that will contribute to lower the energy consumption of buildings and increase their energy flexibility, efficiency, and user satisfaction.

The Deliverable D1.3 identifies the key (functional and non-functional) requirements for the SATO platform considering traditional systems engineering requirement analysis based on SATO's foreseen use cases and the best practices that are already adopted in practical enterprise and scientific environments. This deliverable is a direct output of Task T1.3 (Requirements and System Architecture for the SATO platform and for it to support SRI calculation) from the WP1 (Specification and Requirements for SATO) and will provide guidance for subsequent discussions and decisions that will take place in Deliverables D1.4 (Description of the system architecture of the SATO platform) and all tasks of WP2 (Development of integrated technical Platform for SATO).

The main outcome of this Deliverable D1.3 is the identification and description of the six main functional requirements for the SATO platform and more than a hundred non-functional requirements (divided into twenty-one non-functional criteria). Although this requirement analysis does not prevent new requirements from being identified later during the SATO project, it is comprehensive enough to guide the design and implementation of the SATO platform.

1. Introduction

There is a gap between the predicted and the actual energy consumption in buildings, which is caused mostly due to unrealistic predictions on the occupancy behaviour and the adopted energy management heuristics [1]. Many efforts have been done to profile this gap and increase energy efficiency (e.g., the PROBE studies [2]). The market has reached to a turning point where it must advance beyond simply certifying the energy performance of buildings. The current trend is to continuously assess real-life energy consumption from all energy equipment in a building, allowing to optimise the energy performance of the whole building and its equipment.

The SATO project aims to address this challenge by creating a new *energy self-assessment and optimisation platform* (*SATO platform*) that integrates and monitors energy consumption of building equipment and appliances. This platform will support a self-assessment framework (SAF) built on top of data analysis and machine learning approaches to report energy performance, building behaviour, occupancy, and equipment faults. Aligned with the recent initiative from the EU to introduce a Smart Readiness Indicator (SRI) [3] of buildings, the SAF enables its continuous estimation in useful time for optimising the energy consumption of the monitored buildings.

Additionally, the SATO platform will support the development of Building Information Model (BIM)-based interfaces for analysis and visualisation of the assessments in the various applicable scales and defining locations and specifications of energy consuming equipment, sensors, and actuators into a realistic three-dimensional building model (i.e., BIM). The final goal of the SATO platform is to help demonstrating that these self-assessment and optimisations contribute to optimise the energy management of buildings and increase their energy flexibility, efficiency, and user satisfaction.

1.1. Objective

The main objective of this Deliverable D1.3 is to identify the key (functional and non-functional) requirements for the SATO platform based on SATO's foreseen use cases and the best practices that are already adopted in practical enterprise and scientific environments.

This deliverable is a direct output of Task T1.3 (Requirements and System Architecture for the SATO platform and for it to support SRI calculation) from the WP1 (Specification and Requirements for SATO) and will serve as an input for the Deliverables D1.4 (Description of the system architecture of the SATO platform) and D2.1 (Concept of the SRI enabled SATO platform).

It will guide, together with three other deliverables from WP1 (i.e., D1.1, D1.2, and D1.5), the discussions and decisions in all tasks of WP2 (Development of integrated technical Platform for SATO). WP2 will use the requirements identified in this deliverable to deeply explore the most prominent technologies that will enable the SATO platform to support the requirements.

1.2. Methodology

There are several methodologies for requirement analysis in engineering systems. Some prominent examples include the Systems Engineering Fundamentals from the U.S. Department of Defence [4] and the Functionality, Usability, Reliability, Performance, Supportability (FURPS) model [5]. Additionally, the European Commission (EC) proposed the European Interoperability Framework (EIF) [6], a framework for setting up interoperable digital public services, which provides interesting guidelines for cross-organisational scenarios such as the one of the SATO platform.

In this deliverable, we follow the traditional approach of identifying and separating requirements into two simple categories: functional and non-functional. These classes of requirements are present in most requirement analysis methodologies and provide enough separation for this document. Additionally, we opted by this traditional methodology rather than agile ones because the size and structure of the SATO project favour development cycles associated with the former (i.e., use cases, requirements, design, implement, and evaluate).

We start by briefly describing the foreseen use cases and pilots of the project and identifying the best practices adopted in large-scale interoperable systems. Based on these scenarios and examples, we identify the functional requirements for the SATO platform and derive them into non-functional requirements, by separating the latter into their different criteria.

1.3. Structure of the Document

The remaining of this document is divided into four sections. Section 2 introduces the foreseen use cases and pilots of the SATO project to recall the most important aspects for the requirement analysis that will be delivered by this document. Section 3 presents an overview of several existing initiatives that contribute to the development of interoperable systems and is subdivided into three layers: semantic interoperability, syntactic interoperability, and the technological interoperability. Section 4 derives functional and non-functional requirements for the SATO platform from the foreseen use cases and pilots and from the best practices adopted by existent interoperable initiatives. Finally, Section 5 summarises the identified requirements and concludes this document by providing guidance for the subsequent steps related with the SATO platform.

2. Use Cases and Pilots from the SATO Project

In this section, we present an overview of the foreseen use cases that will help designing the pilot demonstrations within the project. Although the detailed use cases and experiments will be presented in Deliverable D1.5 (Definition of use cases and operational test experiments), some initial cases that were presented in the SATO project proposal are recalled in this section. These use cases and pilots will help to better identify the requirements for the SATO platform and will validate them in the long term.

2.1. SATO Use Cases

The SATO platform intends to enrich the energy management ecosystem with self-assessment and optimisation capabilities (SA&O). There are many use cases where these capabilities can benefit multiple actors and stakeholders, ranging from building managers, energy service providers, and energy aggregators to appliances manufacturers and end-users.

To most of them, the primary goals certainly are to increase energy efficiency and reduce their energy costs. However, other goals are also important, such as: increasing thermal comfort, extending the useful life of appliances, adapt the usage according to shift in energy loads and consumption, integrate multiple hybrid energy systems, monitor the energy impact of individual decisions, demonstrate advanced control strategies, etc. These examples are summarized in the eight use cases (and their goals) presented in Table 1. More detailed descriptions of these use cases will be presented in Deliverable D1.5 of the SATO project.

Table 1: Description of SATO use cases and their goals

Use Case	Description
Case 1	End-users wanting to reduce their energy costs
Case 2	End-user wanting more information about the energy impact of individual decisions
Case 3	A manufacturer of appliances wanting to offer SA&O capabilities

Case 4	An end-user wanting to increase thermal comfort with no additional energy demand
Case 5	A building manager that needs to control hybrid energy systems
Case 6	Energy service provider wants to demonstrate advanced control strategies
Case 7	An energy aggregator wanting to stabilize the electric grid by load shifting
Case 8	An energy service provider wanting to establish a business model for green, local energy for their clients

These general use cases are present in many scenarios from the real world. In the next section, we detail some of the foreseen scenarios that will be explored as project demonstrations in the SATO project.

2.2. SATO Pilots

The SATO project will develop eight pilot demonstrations to attest the benefits of enhancing traditional energy management ecosystem with self-assessment and optimisation. These pilots will include three residential environments, four service environments from universities and municipalities, and four commercial environments from retail stores. They present diverse scales and functionalities, which can be translated into different requirements. Additionally, they encompass three climate regions (Mediterranean, central, and northern Europe), which increases even more the complexity and evaluation of the pilot experiments. Buildings in these pilots contain various types of appliances traditionally present in large, complex smart energy management projects, such as, heat pumps, HVAC systems, lighting, washing machines, dryers, e-charging points, electric vehicles, photovoltaic systems, local energy storage, etc.

Table 2 details the main characteristics of the pilots foreseen in the SATO project with focus on their size, scale on number of devices, the energy management platform deployed on site, the communication directions enabled on the platform and the data delivery methods. As one can observe from the third column, the scale of the pilots ranges from dozens of devices up to thousands. Most pilots will provide Representational State Transfer (REST) APIs for fetching data, while some others will deliver data on Amazon Web Services, which may include the block storage solution S3 (Simple Storage Service) or event-based solutions such as the SQS (Simple Queue Service).

From the foreseen eight pilot demonstrations, one can immediately identify the importance of integrating multiple scenarios, use cases, platforms, and other technologies. With this in mind, we present in the next section an overview of several initiatives that contribute to the creation of interoperable systems in the energy sector and in many other areas.

Table 2: Characteristics of SATO pilots. (* The gateway and/or data delivery method will be defined based on SATO use cases and requirements)

Pilot	Size	Devices	Platform	Communication	Data delivery
Aalborg Residential	2160m²	hundreds	Soft og Teknik proprietary	Bidirectional	REST API
Aalborg University	9000m²	thousands	Schneider	Bidirectional	N/A*

Office			Electric		
Milan Residential	4172m²	dozens	N/A	N/A	N/A
Aspern C4 Technology Centre	6314m²	thousands	Siemens Desigo CC	Bidirectional	AWS S3
Seixal Residential	3451m²	hundreds	EDP re:dy	Bidirectional	AWS S3 or Event-based
Seixal Municipality Office	15000m²	dozens	Sauter GTC	Unidirectional	Modbus
Lisbon Services Building	3745m²	dozens	N/A*	Bidirectional	N/A*
Lisbon/Madrid Retail stores	8034m²	dozens	N/A*	Bidirectional	N/A*

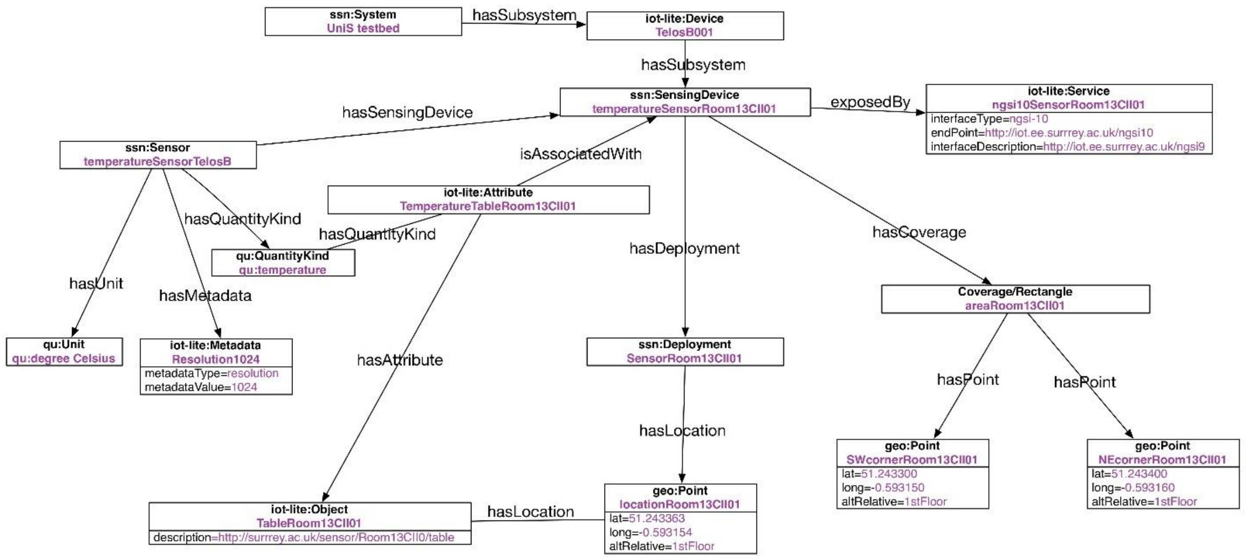
3. Interoperable Initiatives

One of the key aspects the SATO project will focus on, is the integration of heterogeneous energy management systems and devices. Only integrating these elements will make possible for the SATO project to provide the promised self-assessment and optimisation. With this and the European Interoperability Framework (EIF) in mind, we identify three important interoperability layers that directly impact the other requirements for the SATO platform: the semantic interoperability, the syntactic interoperability, and the technical interoperability. We acknowledge interoperability is a non-functional requirement per se, but we opted to start describing here related initiatives using these criteria, that will be revisited in Table 10 (Section 4.2) when we present the specific interoperability requirements for the SATO platform. On each of the interoperability levels, we review the main existing initiatives that are related with the energy management sector and contribute to the deployment of interoperable platforms in this area. These initiatives provide guidance on architectural components since they already implement the best practices in designing and developing large complex system architectures. Additionally, they will be further explored in additional deliverables from the SATO project when architectural decisions must be made.

3.1. Semantic interoperability

Semantics is the study of meaning, while semantic interoperability focuses on adopting a standard collection of concepts and their precise meanings. This common glossary is of extreme importance in large-scale projects because it enables the different components of a system to precisely define their semantics and use a recognized language to communicate with each other, avoiding dubious meanings and confusion. More specifically, in the SATO project, providing semantic interoperability through a common taxonomy makes it easier for components (e.g., devices, services, processes) to communicate with each other [7] [8].

In computer science, semantic knowledge is normally used in the form of an ontology, which is a way of describing concepts (sometimes named classes), the relations between them, their properties,



features, and attributes. A class describes a concept in their domain, where each class can also have subclasses.

Figure 1: Ontology Example - IoT-Lite Ontology [9].

For instance, let us present a basic example of an ontology for IoT devices, the IoT-Lite Ontology [9]. **Error! Reference source not found.** illustrates this ontology, which defines how devices can compose an IoT system, their internal properties, how many and which sensors this device is associated to, as well as their location and covered areas. Adopting an ontology like this enables large platforms to use a common language for describing devices appropriately across the whole platform.

After this brief overview of the ontology concept as the main mean of semantic interoperability, we present three more complex ontologies related with energy management systems that will provide a base ground for the taxonomy used in the whole SATO project.

3.1.1. SAREF

SAREF ontology[1] is a shared model of consensus that facilitates the matching of existing assets in the smart applications domain with energy capabilities. One of its goals is to express the energy context, which is achieved by representing the energy associated characteristics of smart applications. This shared consensus provides a unique representation of the devices and their characteristics, even if their underlying technologies differ from each other. One of the main advantages of SAREF for the energy optimization context is that it was developed with the energy domain in mind due to the involvement of stakeholders from the energy sector [10]. The ontology overview describing the composition of the main classes and relations provided by SAREF is available in Figure 2.

3.1.2. SAREF4ENER

SAREF4ENER[2] is an extension of the SAREF ontology created in collaboration with EEBus[3] (see Section 3.2) and ENERGY@HOME[4], two main stakeholders in the smart energy domain. Its goal is to interconnect different data models [11]. This ontology carries all the benefits of the SAREF ontology

[1] https://saref.etsi.org/
[2] https://saref.etsi.org/saref4ener/v1.1.2/
[3] https://www.eebus.org/
[4] http://www.energy-home.it/

while adding the possibility to support emerging data models. By using SAREF4ENER, devices that support EEBus and Energy@Home data models can easily communicate with each other using any energy management system at home or in the cloud.

Figure 3 and Figure 4 present the two main concepts added by the SAREF4ENER to the SAREF ontology to support the EEBus and ENERGY@HOME data models. They detail how the classes are related to each other and to the SAREF ontology.

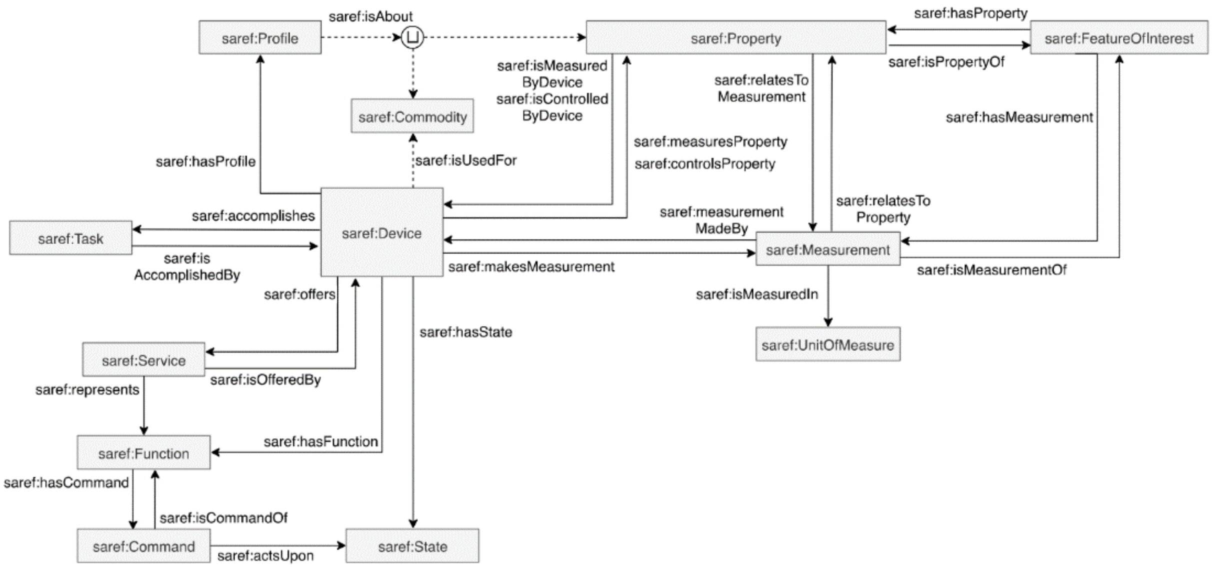


Figure 2: SAREF Ontology Overview [12]

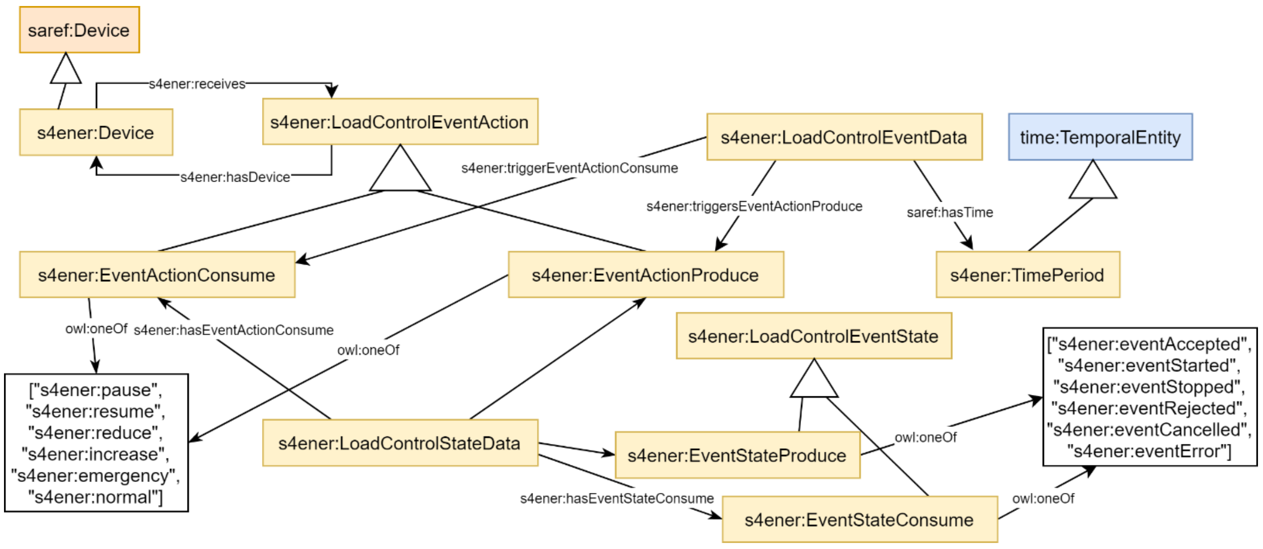

saref:Device
s4ener:receives
s4ener:Device
s4ener:LoadControlEventAction
s4ener:hasDevice
s4ener:LoadControlEventData
time:TemporalEntity
s4ener:triggerEventActionConsume
s4ener:triggersEventActionProduce
saref:hasTime
s4ener:EventActionConsume
s4ener:EventActionProduce
s4ener:TimePeriod
owl:oneOf
s4ener:hasEventActionConsume
owl:oneOf
s4ener:LoadControlEventState
["s4ener:pause", "s4ener:resume", "s4ener:reduce", "s4ener:increase", "s4ener:emergency", "s4ener:normal"]
s4ener:LoadControlStateData
s4ener:EventStateProduce
owl:oneOf
["s4ener:eventAccepted", "s4ener:eventStarted", "s4ener:eventStopped", "s4ener:eventRejected", "s4ener:eventCancelled", "s4ener:eventError"]
s4ener:hasEventStateConsume
s4ener:EventStateConsume
owl:oneOf

Figure 3: SAREF4ENER LoadControlEventData Class [13]

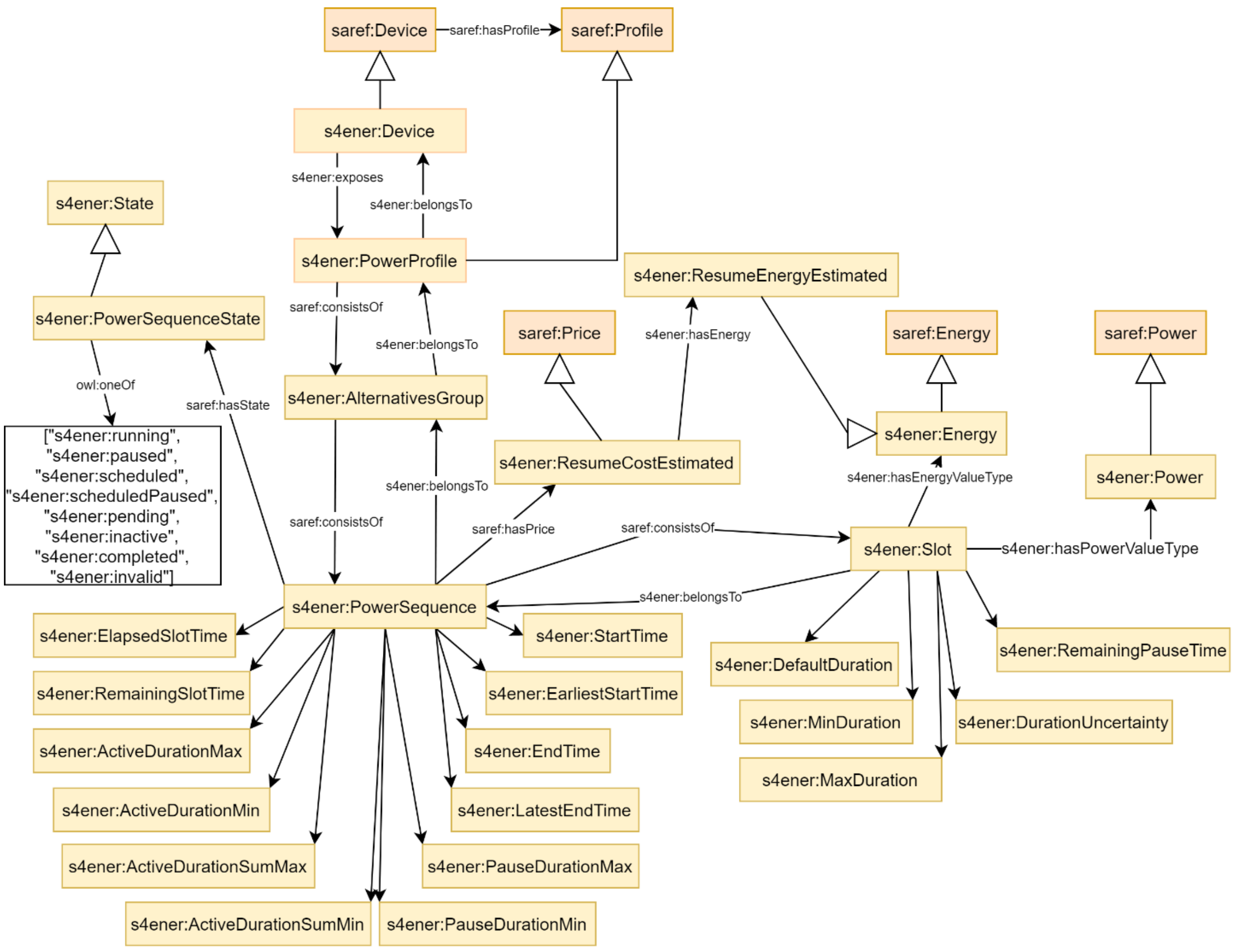


Figure 4: SAREF4ENER PowerProfile Class [13]

3.1.3. SAREF4BLDG

SAREF4BLDG[5] is also an extension of SAREF ontology, similar to the SAREF4ENER. In this case, the extension was created based on the Industry Foundation Classes (IFC) standard for building information, which is a standardized reference model for openBIM data exchange. SAREF4BLDG has the goal to supply the currently missing interoperability among various actors and applications managing building information involved in the different phases of the building life cycle [14]. This goal is achieved by extending some information of the IFC models to include data annotations about smart devices (described in IFC) and their attributes and location in buildings. Figure 5 provides an overview of the whole ontology developed in [15], which places together the diagrams provided in the official documentation.

[5] https://saref.etsi.org/saref4bldg/v1.1.2/

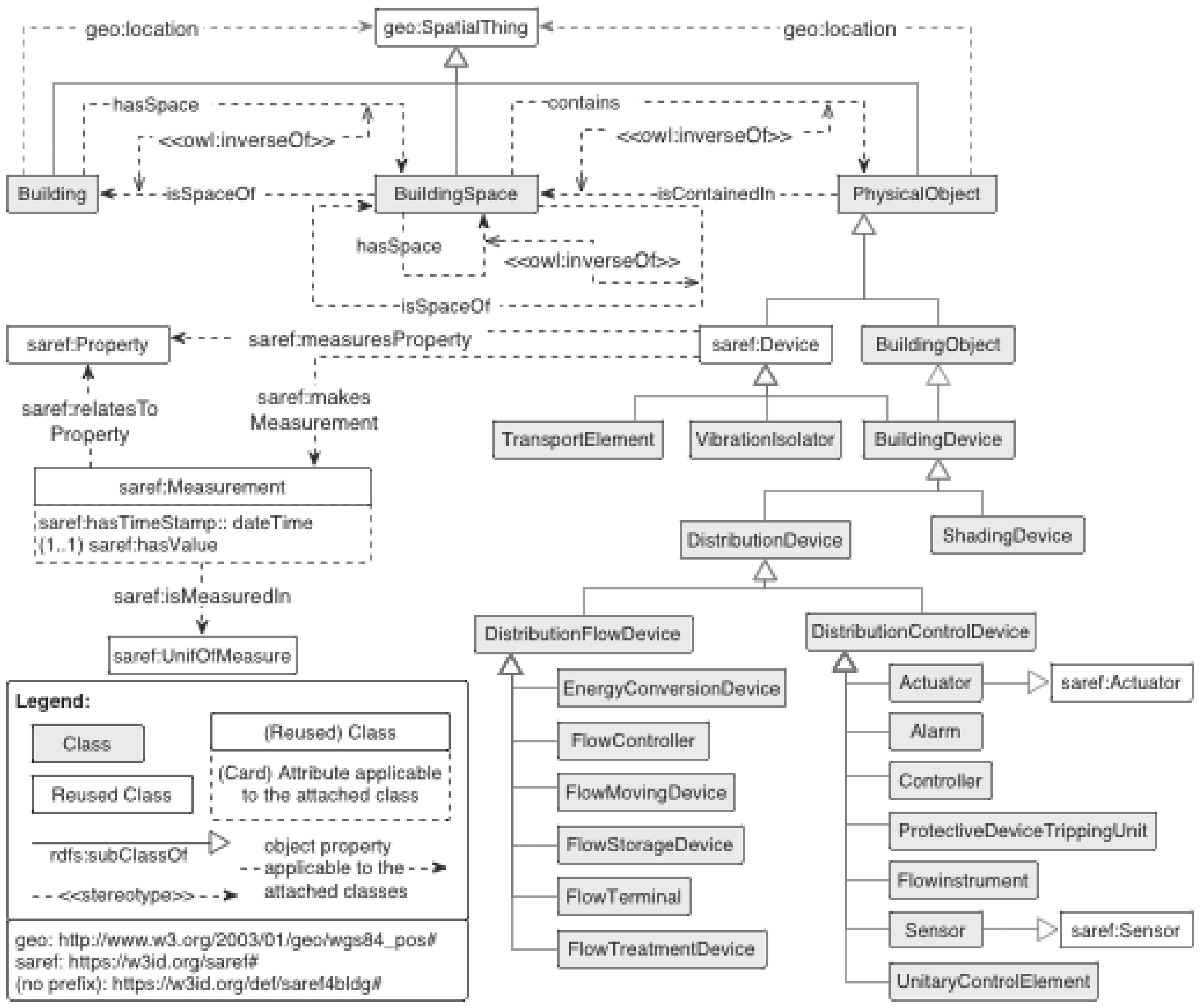


Figure 5: SAREF4BLDG Ontology Overview [15]

3.2. Syntactic interoperability

After defining the semantic interoperability, another important interoperability layer is the syntactic one, which provides a standard representation of the data (and meaning) in the whole system. More specifically, the SATO project intends to integrate heterogeneous devices and multiple external platforms, each of them possibly using their own internal data models. With this complexity in mind, the SATO platform needs to accommodate all the heterogeneous information it receives. One of the most common approaches is to establish a unique global data model at the platform level, in which all components need only to understand this adopted data model since all inbound data will be mapped to it. This solution is widely adopted since it relies on having a component responsible for all complex mappings to a single data model, which prevents the need for mapping to many different proprietary data models and for every component having to know each other. Figure 6 illustrates these two scenarios, with many or a single data model, which vouches for the benefits of having a common data model throughout the SATO platform.

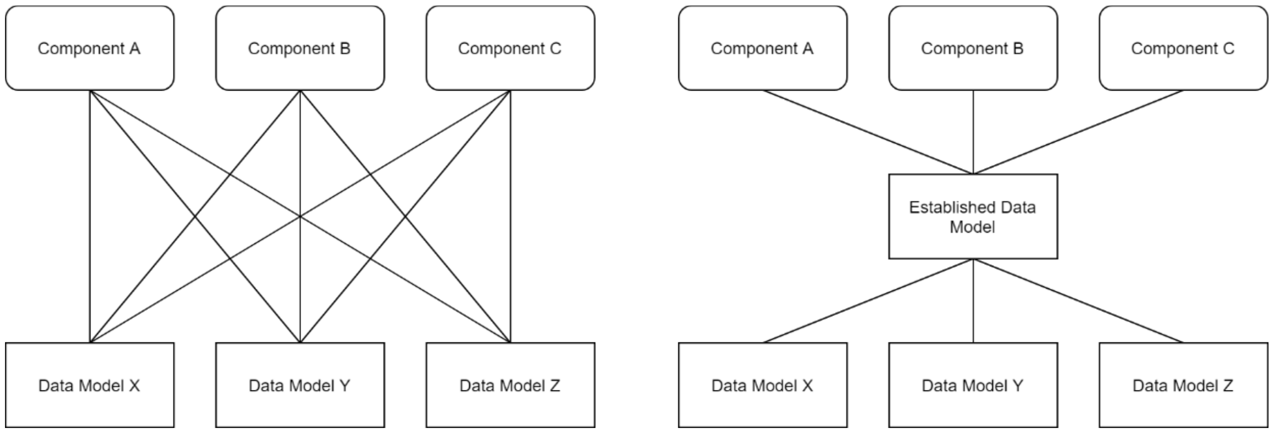


Figure 6: Common Data Model Example

This common data model can support multiple completeness levels, in which a first level introduces minimal required information and metadata, while it can be later enhanced with semantic ontologies and more detailed information from other services. For example, sample and measurement data might not need to be sent to the platform using the same data model, but it can be enriched with this adopted data model to facilitate the self-assessment about buildings' energy consumption.

In the following subsections we review the main existing initiatives related to the energy management in buildings, that may be useful to promote the syntactic interoperability of the SATO platform.

3.2.1. EEBus

EEBUS is a family of protocols for the Internet of Things. It is composed of EEBUS SPINE and EEBUS SHIP and has the goal of creating a common non-proprietary language for energy. Currently EEBus is already implemented by some commercial platforms (e.g., Bosch IoT Gateway Software [16]), where SHIP and SPINE protocols are supported, making devices to be easily integrated on it. Figure 7 describes the architecture of an EEBus system with a clear separation of layers and their available solutions.

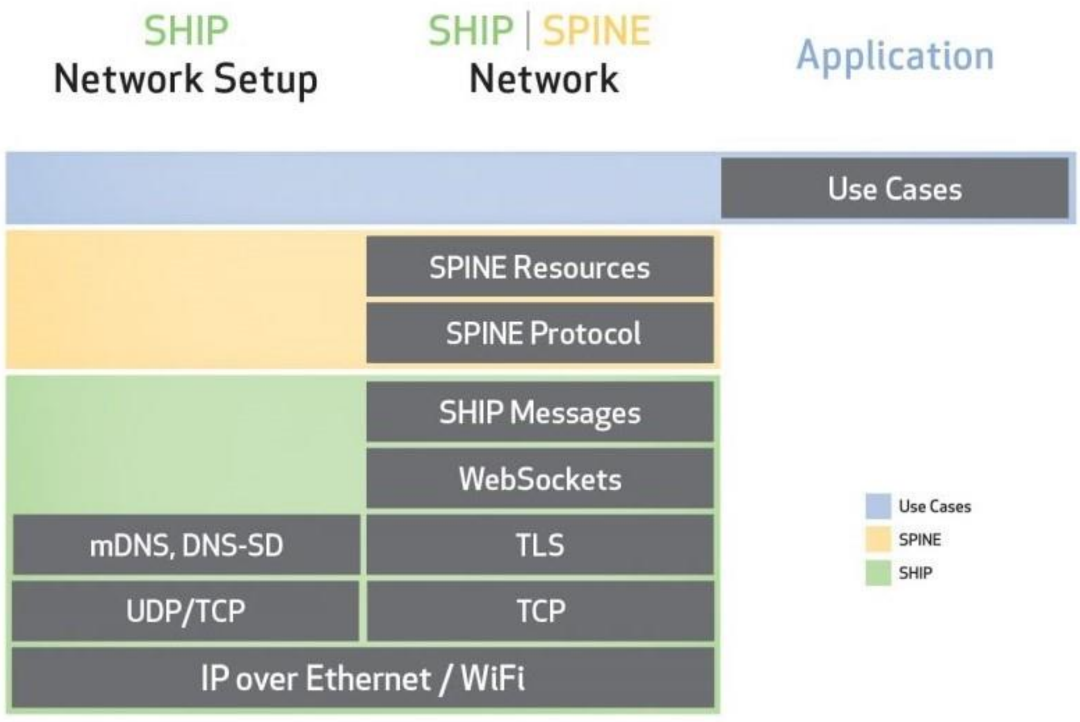


Figure 7: EEBus System Architecture [17]

SHIP
SHIP [18] (Smart Home IP) is an IP-based approach for interoperable connectivity of smart home appliances. It is used between the Network Layer and Application Layer based on the OSI model. It can use TCP or UDP protocols for transporting messages at the Communication Layer. It was developed to be used as a common communication technology, handling all the complexities of establishing a secure, reliable communication between devices [19]. Additionally, it offers diverse mechanisms for registering the SHIP nodes, which cover many practical scenarios.

SPINE
EEBUS SPINE [20] (Smart Premises Interoperable Neutral-Message) establishes a common application-level protocol (OSI Layer 7) by specifying the format of messages to be used by protocols [20] (SPINE defines them as datagrams). The information it contains follows the resource specification from SPINE [20], which contains different data models (focusing on the smart energy domain) to be used in the message exchange. These two components enable a standardization in the field of smart energy, but it expects that every device can talk according to the specification or at least support the mapping between the proprietary protocol and SPINE. This second scenario allows one to integrate non-SPINE devices with SPINE compatible systems.

In a more specific perspective, the SPINE protocol describes also how a device should be represented in a standard representation, which grants the service discovery mechanisms explained in their specification. This specification also provides details on how to add other mechanisms to the message communication, such as binding, subscription, or the use of different communication modes. SPINE messages can be exchanged directly between two devices or by using a third device that can act as a middleman when the edge devices do not know the location of each other. SPINE Devices can also describe the use cases supported by them according to the use of original cases provided by EEBus. Its usage is not restricted to SHIP transport since it can be transmitted by any TCP or UDP communication protocol. Finally, messages are originally specified using XML schemas (XSD), however JSON or any other format can be used to represent information in the system.

EEBus also provides several use cases, which are manuals on how to use the modular, universal SPINE resources to easily implement customised concrete solutions fast. This way, EEBus (SPINE and SHIP) provides interoperable, expandable, backward-compatible interfaces. Use cases can be discovered [20] similarly to the mechanism of the device service discovery mentioned before, where devices can announce which use cases they support due to the common standard used.

3.2.2. oBix

oBIX [21] (Open Building Information Xchange) is a standard for REST Web Services and XML for building control systems. It is designed to provide access to the embedded software systems which sense and control the world around us [22]. The design philosophy of oBIX is based on a small, extensible data model that maps it to a simple fixed XML syntax. Used as a common data model it has the advantage of not being very strict in the data model composition when compared to other specifications. However, the scope of the SATO project may render this simplicity as a disadvantage since the project needs some data model which covers the energy domain, where the data representation is (although extensible) well defined. Another disadvantage is the fact that oBIX is an outdated specification, which was created more than a decade ago (last updated in 2013). Technology evolves rapidly which may render this specification inadequate given the emerging alternative technologies that can be used in the SATO project [22].

3.2.3. CIM

CIM (Common Information Model) is an open standard that defines how to manage computing devices and the relations between them. It is based on object-oriented programming model and uses these techniques to represent an enterprise. It is composed of a specification, an extensible Schema, and a metamodel. CIM Management Schema is the building-block for management platforms and applications model of the descriptions it represents the object and their relations and the bases for their management. CIM structures the environment as a collection of interrelated systems, each composed of discrete elements [23]. CIM Specification defines the details for the integration with other management tools (based on UML) and the language in which the CIM Schema is defined. CIM Metamodel defines the semantics for the construction of new conformant models and the schema that represents those models [23] [24].

3.2.4. Other initiatives

In the area related to BIM technology, OpenBIM promotes the use of open standards to facilitate the interoperability between different software solutions. Industry Foundation Classes (IFC) is a standardized, digital description of the built asset industry. It is an open, international standard (ISO16739-1:2018) and promotes vendor-neutral, or agnostic, and usable capabilities across a wide range of hardware devices, software platforms, and interfaces for many different use cases [25].

GL Transmission Format (GlTF) is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by engines and applications. The GlTF minimizes the size of 3D assets, and the runtime processing needed to unpack and use them. It defines an extensible, publishing format that streamlines authoring workflows and interactive services by enabling the interoperable use of 3D content across the industry [26].

3.3. Technological interoperability

Technological interoperability is an abstract concept that focuses on the seamless interaction between many diverse components within a system. It adopts standard approaches to enable components working together to achieve a global purpose, where they collaborate even if each component has a different specific goal or uses diverse internal mechanisms.

For the SATO platform, a concrete example that requires technical interoperability is the integration of heterogeneous external platforms and devices into a coherent vision of all the entities that compose the building energy management ecosystem. These diverse solutions must interact with the SATO platform through standard interfaces and pathways. Additionally, the components from the SATO platform must support the same communication approach and be agnostic on where they are deployed (e.g., in a public or private cloud) since external platforms may be plugged into the SATO platform and provide data in diverse ways and locations that need to be standardised by the SATO connectors.

In the following sections, we introduce some existing solutions that promote the best practices in technological interoperability for IoT and energy management systems, which is of extreme importance for the development of SATO. We start by introducing some platforms that integrate multiple heterogeneous legacy and new devices. Then, we present solutions that enable event-based communication for multiple platforms. Finally, we introduce some initiatives that provide containerisation, which is a lightweight type of virtualisation that enables the seamless deployment of components and services in various computing scenarios and scales.

3.3.1. Platforms for integrating devices

When using devices from existing platforms the integration should be easy since they already have their gateway locally or at their cloud, allowing for their devices to access remotely (given the proper permissions). The necessary adaptation to the SATO platform would be to comprehend the data models used from each of the platforms to integrate them into the SATO project.

Legacy Devices
Some of the existing buildings already have their legacy systems, which might not be supported from anywhere in the cloud, only doing the communication locally which hinders the coverage by the SATO platform. Regarding the support of the platform to legacy devices or systems, there are two main solutions, which are the following:

- Directly supporting every legacy equipment individually.
- Supporting the whole set of devices through a gateway.

When considering the former, one needs to support the mapping to a big number of data models even if using a common data model at the level of the SATO platform. To achieve this direct connection to the platform, every device must support a connection to the cloud, which some devices may not support without the help of a gateway. The main advantage of this solution would be that there would be no need to integrate a gateway at the deployment location.

The second solution requires the deployment of a gateway within building’s boundaries with the respective software to locally integrate the devices and only later integrate them to the platform. This apparent drawback comes with many benefits such as, the gateway can translate the used data models to the adopted data model of SATO, which would reduce drastically the number of mappings required for the SATO platform. It also improves the security of the information passing through the network since gateways normally have better computing power than edge devices and provide the support needed to connect these edge devices to the cloud. This solution may require a nonnegligible computing effort from the gateway if it is responsible for a deployment with a very large number of devices (and their respective events). Finally, this solution also reduces some load in the network since the gateway can aggregate information and use better serialization techniques to reduce the amount of data being forwarded to the SATO platform.

By comparing the two options, it is possible to note that one’s advantages are the other’s disadvantages, which require trading-off the pros and cons of each scenario, which will be done in subsequent tasks in the project.

FIWARE
FIWARE is an IoT platform that enables the integration of diverse IoT devices in a single platform [27]. Its main characteristic is the use of the NGSI protocol to enable such integration. FIWARE’s main component is the *Orion Context Broker,* where a REST API is exposed to manage the devices

integrated into the platform, with the *IoT Agents* being the responsible entities for communicating with and integrating the devices. This integration is achieved by mapping the device's attributes and characteristics in the Agent itself and with the logical device represented at the Orion Context Broker. Figure 8 depicts a simplified view of the general architecture of FIWARE.

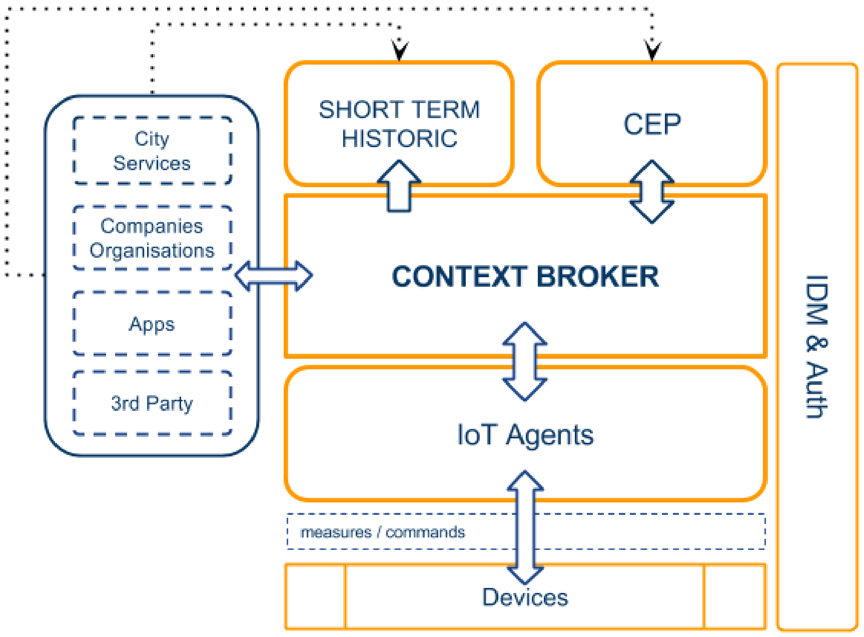


Figure 8: FIWARE Example Architecture [28]

When using the IoT Agents mapping, FIWARE does not follow a strict common data model since devices with the same attributes or characteristics can have different mappings. This choice can be difficult to hold in a system where millions of devices are integrated and where data analysis and assessments need to be made as fast as the expected frequency in the SATO platform. One may consider adapting the previously explained solution (i.e., using the IoT Agents) to a scenario with a more standard data model.

Communication between the components of FIWARE is done through direct point-to-point channels, which may be inviable for the SATO platform due to the need to support event-based streaming communications. Another issue is that, at the time of writing, FIWARE apparently lacks other types of device authentication beyond the API key methodology. However, since it is an open-source solution, one can modify or extend their solution to fit into other requirements (although this adaptation can be somewhat cumbersome given the FIWARE dimension).

OGEMA

OGEMA is another open-source software platform that provides a hardware-independent execution environment for energy management applications. It was designed to serve as a gateway allowing applications to access different types of connected hardware [29]. This is achieved with the usage of drivers that take care of device specific complexity. Figure 9 provides an overview of the OGEMA architecture, which enables the analysis of its main components.

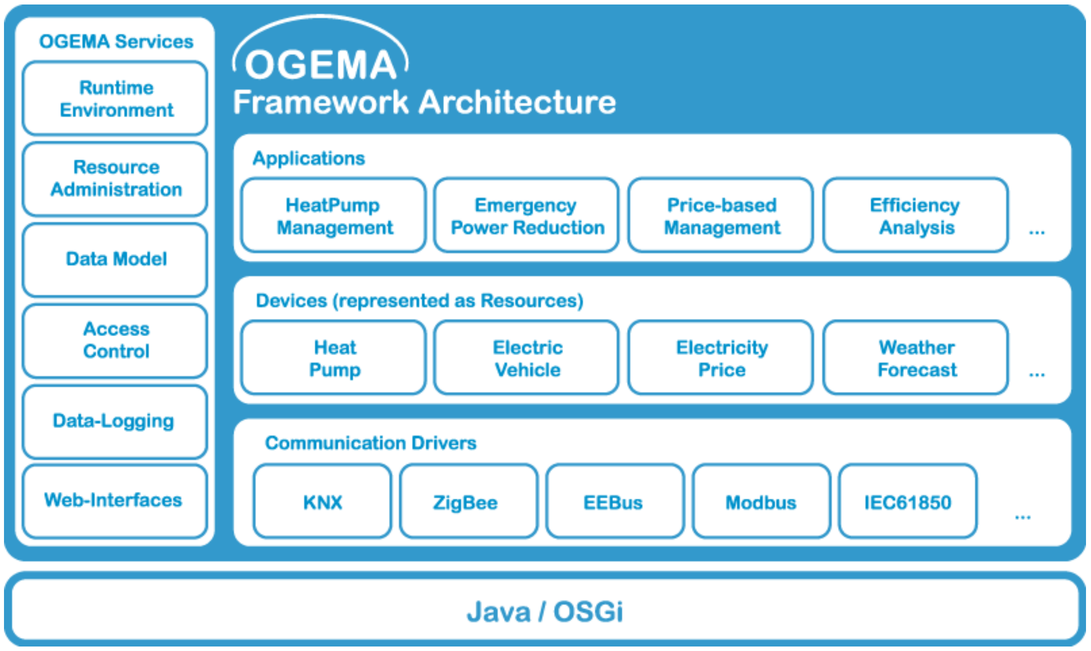


Figure 9: OGEMA Framework Architecture [30]

OGEMA provides access to external devices via traditional REST interfaces considering standardised data models and device services. From the information provided, the main goal is to be used as a gateway for devices in a physical use case, which can be useful for legacy equipment where there is no gateway platform in place. OGEMA can forward data from devices through an OGEMA gateway API in their standardized data models [22]. This feature grants, to heterogeneous legacy devices, access to the SATO platform, which is important for the SATO project.

EDP RE:DY

This platform is an energy management solution provided by the Portuguese company EDP, a partner of SATO project, enabling customers to manage the energy consumption of their appliances, and receive real-time data in the web or smartphone [31].

The EDP re:dy system is composed by an EDP re:dy box and multiple IoT peripherals. The EDP re:dy box is the main component of the system since it sends (resp. receives) data to (resp. from) the peripherals and acts as a gateway. It enables these heterogeneous devices to interact with the cloud, as any other IoT platform. The EDP re:dy is one of the external platforms which the SATO platform will have to communicate with since the former will be used in the buildings of some foreseen project pilots.

Siemens Navigator and Desigo CC

Siemens Navigator is a cloud-based energy and asset management platform that serves as the foundation for a comprehensive set of services offerings. It centralizes disparate data sets related with building's performance, energy consumption, environmental conditions, assets' health, equipment features, real-time performance, and maintenance tasks (e.g., schedules or work orders). With these data sets, the platform generates results through its services improving the environment conditions, increasing operational efficiency, and reducing the risks associated with downtime from equipment failures [32].

Siemens Desigo CC is another powerful platform for high-performing building management. It is open-by-design and integrates many device categories on a single platform: from lighting, power, security, fire safety, heating, ventilation, and air conditioning to third-party integration. In short, it helps different stakeholders to easily manage the building or a space, even when they are composed of different systems. It provides a platform that enables the communication and interoperation between these spaces and devices via a centralized command and control centre [33].

These two platforms from Siemens are also present in some foreseen project pilots and, for that reason, will be integrated with the SATO platform.

3.3.2. Messaging System

Previous discussions reinforce the importance of the SATO platform supporting numerous heterogenous components and platforms interacting with each other. To facilitate the development of each component and making it as independent as possible, the SATO platform may adopt an event-based communication model by employing a streaming system to manage message delivery under a publish-subscribe model. This model also enables the platform supporting multiple data flows at the same time.

This communication model decouples the platform components from the communication management. However, it implies that this communication system must be available and reliable since all the components (and consequently the whole platform) depends on it, while it must avoid becoming a bottleneck on the system.

Another objective with this component is to make the platform more compliant and dynamic with the separation of the control flow from the data flow. The former represents messages that are used to manage the system (e.g., adding or removing devices from the platform), while the latter represents the data measurements sent by the devices.

Apache Kafka
Apache Kafka is a distributed messaging system based on the publisher-subscriber paradigm. Its main goals are to provide low overhead for network communication and storage, high throughput in supporting lots of messages for publishing and subscribing, guaranteeing fault-tolerance in case of failures in some servers, among others. Its architecture was designed for a scalable low-latency distribution and delivery of messages, while providing APIs for the Consumer, Producer, and Admin users [34].

Kafka is composed of several parts: a broker that handles all requests from clients and keeps data replicated within the cluster, a Zookeeper[6] coordination service for handling the consistency of the system state, the producer that produces events and send them to the broker, and the consumer that consumes events from the *broker.*

Apache Pulsar
Apache Pulsar is another high-performance solution for server-to-server messaging. It has some interesting features such as, the native support for multiple clusters with seamless geo-replication of messages across clusters, low latency when publishing events, scalability to over than a million topics, client APIs for diverse programming languages (e.g., Java, Go, Python and C++), and multiple subscription modes (e.g., exclusive, shared, and failover) for topics [35].

Pulsar is also built on top of the publish-subscribe paradigm, where producers create *messages* and publish them. The subscribers are notified of new *messages* and consume them. The Pulsar messaging system is composed of a broker that exposes a REST interface administration, *topic* lookup, and a dispatcher that handles all Pulsar message transfers [36]. Pulse also uses Zookeeper to handle the state of the cluster and provides a *b*ookkeeper that handles the persistent storage of messages.

[6] https://zookeeper.apache.org/

NATS
NATS is an open-source high performance, lightweight, cloud native, messaging system that follows the at-most-once and at-least-once delivery. It provides availability, is easy to use, and supports observable and scalable services and event/data streams [37].

Since it was developed with cloud computing in mind, it can be integrated with Kubernetes and Prometheus. It supports the publish-subscribe pattern, as well as Request-Reply and Queue Groups.

RABBITMQ
RabbitMQ is a lightweight easy-to-deploy message broker that was designed with interoperability, performance, and stability as their main goals. In terms of reliability, it offers reliable delivery and publisher confirmation. It also provides a large library of clients that are written in various languages [38]. RabbitMQ supports both Classic Mirrored Queues, Quorum Queues, and Priority Queues. These implementations position RabbitMQ as a very dynamic solution that can fit the requirements for the SATO platform.

Eclipse Mosquitto
Eclipse Mosquitto is also a message broker, but it implements the MQTT protocol. It is also based on the publish/subscribe model like Kafka, but it is not limited to this paradigm. Its focus is on storing large amounts of data in disks and allowing consumption in real-time. It was also designed to be a cluster with multiple nodes and to scale horizontally. Another alternative is to explore the idea of using Kafka with a message broker (e.g., HiveMQ, Mosquitto) [39].

3.3.3. Containers

A container is a runtime environment that packages all the code and dependencies in a way that the app can reliably run agnostic of the computing platform it is running on. It can be seen as a bundle of the application with the code, the runtime, system tools, the libraries, and its settings [40]. There are several technologies providing containers, where Docker is one of the most popular. Other alternatives include Windows Containers, LXD, and podman.

The use of containers allows to separate the applications from the infrastructure while not being a virtual machine. A virtual machine is an emulation of the computer system. For instance, it allows to simulate two separate computers on the hardware of only one [41]. A container, instead of virtualizing the computer, only virtualizes the Operating system, this makes it much lighter than a Virtual machine [42].

Another important tool associated with containers is the process of scheduling the work of individual containers, which allows one to automate various processes in deploying, managing, and scaling containerized applications [43]. Common managers include Kubernetes, Docker Swarm, and Red Hat OpenShift. They allow the development of cloud-native applications agnostic of the cloud provider.

4. Requirements for the SATO platform

The use cases and pilots foreseen in the SATO project (Section 2) motivate the scenarios that the SATO platform will benefit the most. Additionally, they validate the requirement analysis that will be presented in this section. Notwithstanding, as mentioned in Section 3, interoperability is one of the key challenges the SATO platform must deal to enable all the foreseen use cases and pilots since they will require the integration of diverse devices and external platforms.

The present section identifies and details the main functional and non-functional requirements for the SATO platform based on traditional approaches for requirement analysis, on the previous experiences of the SATO team, on related works, and on the analysis that resulted in Sections 2 and 3. This result is the main output of this document and will guide the design and development of the SATO platform and its components.

4.1. Functional Requirements

In this section, we summarise the functional requirements for the SATO platform coming from the use cases and pilots. The uttermost goal of the SATO platform is to enable the self-assessment framework and self-optimisation services, which are the entities that will consummate the benefits of adopting the solutions from the SATO project as the buildings' energy management system.

The first functional requirement (*F-1*) for the SATO platform requires it to support the mentioned self-assessment framework. Note that the specific requirements for the self-assessment framework will be described in the coming Deliverable D1.2.

F-1. Enable the self-assessment of buildings' energy consumption and production.

By aggregating information about all energy-consuming devices from buildings, the SATO platform will also enable the self-assessment framework to calculate and certify an important recently introduced indicator for the energy management ecosystem: the SRI of a building. This functional requirement (*F-2*) will only be possible if the SATO platform is holistically aware of the buildings' features and devices' specific capabilities.

F-2. Enable the dynamic assessment of buildings' SRI (Smart Readiness Indicator).

Energy-consuming devices will provide measurements to the SATO platform, which will prepare them for later processing in the self-assessment framework. The results from this framework will empower end-users and stakeholders to identify opportunities for optimising the energy resources of the buildings connected to the platform. The third functional requirement (*F-3*) refers exactly to this support from the SATO platform to the optimisation process by collecting and protecting data and intermediating control operations to actuators.

F-3. Enable optimising buildings' energy resources.

The fourth functional requirement (*F-4*) is more specific to the heterogeneous nature of the devices present in the buildings, which must be handled by the SATO platform considering standard approaches and solutions.

F-4. Enable the integrated management of dispersed and diverse devices.

The SATO project will provide a holistic view of the buildings' elements in a standard visualisation and control tool through a BIM. The fifth functional requirement (*F-5*) for the SATO platform is associated with the awareness of entities' location within buildings.

F-5. Enable a location-aware visualisation of the assessments.

Finally, the sixth functional requirement (*F-6*) is associated with the ability to incorporate and deal with user preferences. The SATO platform must be aware of these customised preferences and take decisions accordingly.

F-6. Incorporate knowledge from user preferences.

These six functional requirements represent the main features the SATO platform must support or enable from upper layers within the SATO project. To make the platform capable of accomplishing them, we identify (in Section 4.2) the non-functional requirements that can be extracted from the mentioned functional requirements, use cases and pilots.

4.2. Non-Functional Requirements

In this section, we extend the functional requirements from the previous section into several non-functional requirements that will guide the design and the development of the SATO platform in the months to come.

Before advancing to the non-functional requirements, it is useful to characterize the components of the SATO platform. Components are *processes* (i.e., processing resources), *storage* systems (i.e., storage resources), or *services* (i.e., processing, storage, and network resources). The platform considers *Devices* and *Gateways* from buildings as external components. In Figure 10, these components are interconnected through *network* and *communication* services, where the thinnest lines refer to the flow of control data and commands (e.g., adding a device to the platform or sending actuation commands) and the thickest lines refer to the flow of data and measurements sent by the devices. Services and storage components must be available and tolerate faults caused by crashes, omissions, or network delays in requests and responses. Figure 10 presents a bottom-up overview of how these components are distributed across the layers of the SATO project, including the SATO platform, the self-assessment processes, and the self-optimisation ones.

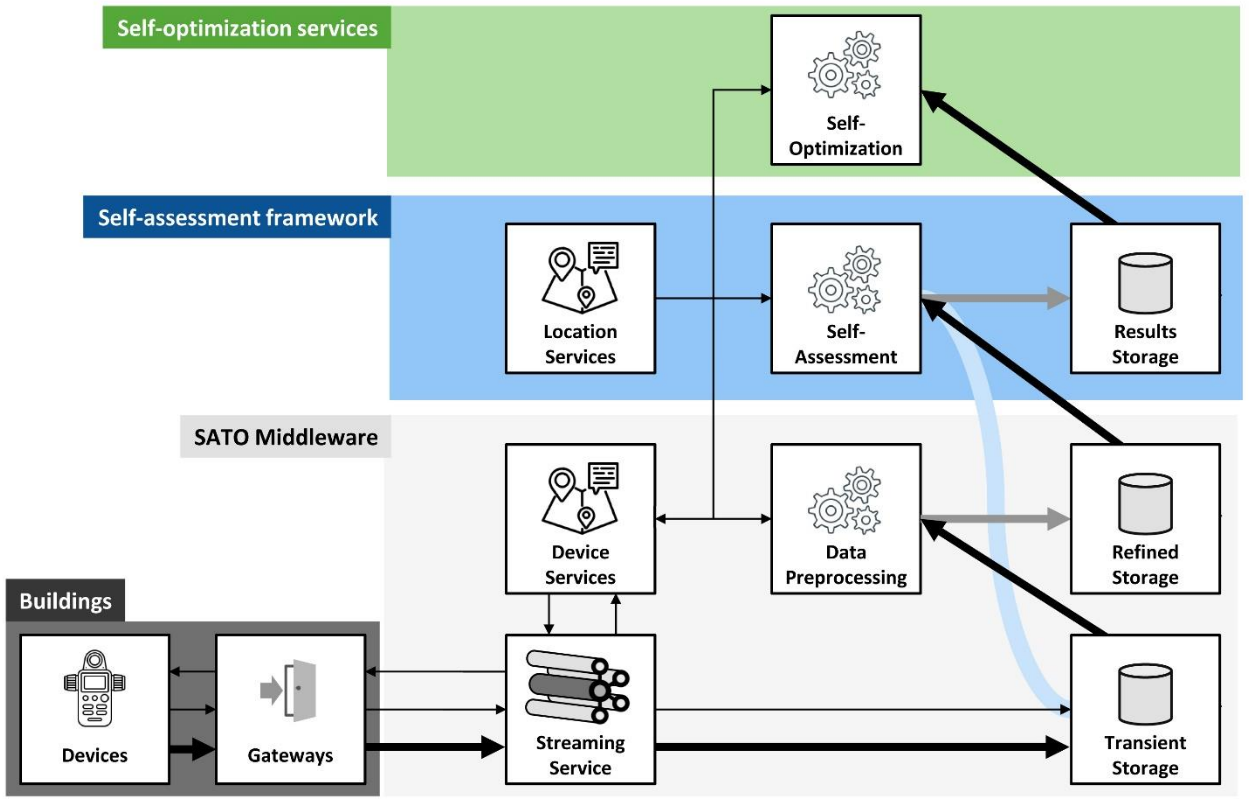


Figure 10: A bottom-up overview of the components of the SATO project

Devices represent all the possible heterogeneous devices that can be present in a building, which are the components responsible for providing data and measurements to the SATO platform. *Gateways* aggregate data from multiple devices from buildings or directly forward it to the SATO platform. They are also responsible for managing and interfacing these devices using lower-level communication and actuation protocols.

The *SATO platform* is a middleware component that ingests data and control commands from buildings as fast as possible. A *Streaming Service* is responsible for receiving all these data points considering an event-based streaming that is delivered to the Device Services and the Transient Storage. The *Transient Storage* is an intermediate persistent storage component that quickly stores the events received by the platform and provides them to the subsequent processes. The *Device Services* is a service that aggregates and manages all the information (e.g., features, protocols) related with the devices existent on each building connected to the SATO platform and provides it to subsequent processes using standard data models that semantically conform with standard device taxonomy and ontologies. The *Data Pre-processing* step is responsible for preparing the data for future processing by adding metadata to data and enhancing data quality and semantics, which is later persistently stored on the Refined Storage. The *Refined Storage* safely stores the enriched, collected data for the processing steps that will happen in the upper layers of the SATO architecture.

The *Self-Assessment Framework* is an intermediate layer that encompasses the *Self-Assessment* processing component, which will perform the core computations to provide the certification, categorisation, labelling, and the assessment of other building and equipment/appliances performance indicators (e.g., the SRI) for the SATO project. Additionally, this layer contains the *Location Services*, which manages and provides all the semantic information related with the physical attributes and disposition of buildings and how the devices are distributed on them. The *Results Storage* component is responsible for storing the output of all the self-assessment processes, which will serve as input for the upper optimisation layer.

The *Self-Optimisation Services* is a layer that encompass the *Self-Optimisation* process responsible for taking decisions based on the assessments, optimising the whole energy consumption of the buildings connected to the SATO platform, and controlling through commands and actuations the devices present on the buildings.

The remaining of this section is divided into twenty-one topics, each of them representing a traditional non-functional criterion that is translated into several non-functional requirements specific for the SATO platform. These criteria are the main aspects that are identified by traditional requirement analysis from systems engineering.

Dependability
Dependability is an integrative concept that guarantees the delivery of services in a trustworthy manner despite the existence of failures. The focus on this criterion is to achieve fault tolerance as a means of survivability by masking faults that could compromise the trust in the system. Additionally, resilience will be considered to provide a persistent dependability even in the presence of changes in the system. The SATO platform should be able to maintain its operational status, even when there are changes, without any major disruption that should not affect buildings, occupants, users, stakeholders, and their data. This criterion results into the first seven identified non-functional requirements, which are described in Table 3.

Table 3: Non-functional requirements from the Dependability criterion

ID	Requirement	Priority
N-A1	The platform shall tolerate faults in a portion of its components	1
N-A2	Crash faults shall be masked	2
N-A3	Communication channels shall be reliable	1
N-A4	Devices shall be protected against tampering	2
N-A5	The platform shall provide ways to ensure resilience	2
N-A6	Components should be monitored and patched for known vulnerabilities	1
N-A7	The platform shall separate control and data planes for data communication	1

Confidentiality
Confidentiality is one of the pillars of security since it guarantees the semantic protection of data flowing (i.e., data in transit) and being stored (i.e., data at rest) in the whole SATO platform. It protects collected data from malicious users trying to read privacy-sensitive data as well as honest-but-curious entities that may cause (un)intentional data leakages. Mechanisms guaranteeing confidentiality together with authentication and authorization means, provide complete access control for confidential data. This criterion results into five specific non-functional requirements for the SATO platform, which are presented in Table 4.

Table 4: Non-functional requirements from the Confidentiality criterion

ID	Requirement	Priority
N-B1	Communication with external platforms shall be encrypted	1
N-B2	Communication within the platform shall apply appropriate security measures	2
N-B3	The platform shall provide secure storage resources	1
N-B4	Personal data in storage shall be ciphered	1
N-B5	Secure storage of data shall be ensured	1

Integrity
Integrity is the second pillar of the security triad and guarantees that received or stored data was not tampered, modified, or lost in transit or at rest by (un)intentional actors or hardware and software failures. This criterion consists of three non-functional requirements, which are described in Table 5.

Table 5: Non-functional requirements from the Integrity criterion

ID	Requirement	Priority
N-C1	Communication shall protect data integrity	1
N-C2	Persistent storage shall support integrity checks	2
N-C3	Gateway integrity check must be supported	2

Availability
Data availability completes the security triad by guaranteeing that information is accessible to authorised users. It usually refers to reliability and system uptime and can be affected by a series of faults (e.g., a crash on a server, or a connection lost). This criterion motivates the definition of the three non-functional requirements enlisted on Table 6.

Table 6: Non-functional requirements from the Availability criterion

ID	Requirement	Priority
N-D1	The platform shall maintain available its communication	1
N-D2	Internal services shall be highly available	1
N-D3	Continuous connectivity should be supported	1

Privacy
Privacy is another non-functional criterion related with security and data protection. Collected data may have different privacy-sensitivity levels that require appropriate security measures. With the tightening of privacy-related legislation (e.g., General Data Protection Regulation - GDPR) and the increasing severity of data breaches and leakages, the SATO project must enforce the best practices in anonymizing identifiable data such as names, personal documents, and even device identification properties (e.g., MAC addresses). This non-functional criterion results in the seven specific non-functional requirements present in Table 7.

Table 7: Non-functional requirements from the Privacy criterion

ID	Requirement	Priority
N-E1	Processing shall take place on trusted nodes	1
N-E2	Privacy-sensitive data should be anonymized when possible	1
N-E3	Privacy-sensitive data shall be kept safe	1
N-E4	Building occupants will not be personally identified under any circumstances	1
N-E5	User anonymity must be ensured at communication level	1
N-E6	User or device shall be anonymised (or identified by a pseudonym)	1
N-E7	The identifier of the device (ID of an RFID tag or MAC address of Wireless Sensor for example) must not be tracked by unauthorized entities	1

Access Control
Authentication and authorisation complement the confidentiality to provide access control means to the data in transit and at rest. Restricting access to data increases the trust on the systems, which is an important aspect for the SATO platform in the energy management process. This criterion was translated into thirteen non-functional requirements presented in Table 8.

Table 8: Non-functional requirements from the Access Control criterion

ID	Requirement	Priority
N-F1	Authentication shall be carried out before devices delivering data	1
N-F2	Only authenticated devices shall provide data for the platform	1
N-F3	Only authenticated external platforms shall insert or remove devices from the platform	1
N-F4	Only authorized components access and process data	1

N-F5	The platform shall support context-aware access policies	2
N-F6	Data owners shall be able to set access-control rights/ policies to their data stored on resources	2
N-F7	User's data shall be processed only with the user's consent	1
N-F8	User shall have the right to transfer his personal data from a service to another	2
N-F9	Applications shall be signed and checked	1
N-F10	Authentication of a user shall be carried out to access a service	1
N-F11	Access-control rights/ policies (set up by data owners) shall not be published publicly	2
N-F12	Platform shall provide an authorization policy to service for the different users or devices	2
N-F13	Components shall be able to evaluate access requests depending on access control policies	2

Persistence

Data in transit may be lost before arriving at the destination or data at rest may be lost before being processed. For those reasons, the SATO platform must provide persistence for its events. This criterion results into two non-functional requirements, which are presented in Table 9.

Table 9: Non-functional requirements from the Durability/Persistence criterion

ID	Requirement	Priority
N-G1	Events shall be kept in the communications middleware until they are persisted in storage or received by the destination	1
N-G2	Data shall be persistently stored as soon as possible	2

Interoperability

Interoperability is a key requirement for the SATO platform, since it will integrate heterogeneous devices and external platforms. Section 3 already described several important aspects about interoperability and its internal layers. This criterion results in ten non-functional requirements that are shown in Table 10.

Table 10: Non-functional requirements from the Interoperability criterion

ID	Requirement	Priority
N-H1	Interconnectivity shall be based on a message brokering layer	1
N-H2	The platform shall interconnect different external commercial platforms	1
N-H3	Devices shall announce themselves to an intermediate gateway or directly to the platform	1
N-H4	External commercial platforms shall manage devices that will be associated to SATO through gateways	1
N-H5	Platform shall support newly deployed and already available legacy systems	1
N-H6	All components shall use the same data models to represent devices and locations	1
N-H7	Data model shall natively support the interconnection and composition of devices and systems produced by different manufacturers	2
N-H8	Services shall provide APIs for other components	1
N-H9	The platform shall support bidirectional communication (i.e., inbound data and outbound actuation)	2
N-H10	Components shall consider standard Information Modelling processes	1

Auditability

Auditability refers to the systematic ability of detecting and reporting actions that happened in the system. It is an important step to detect data incidents, analyse leakages, and sanction misuses. The four non-functional requirements derived from this criterion are presented in Table 11.

Table 11: Non-functional requirements from the Auditability criterion

ID	Requirement	Priority
N-I1	Security incidents shall be logged	1
N-I2	Summaries of energy-consumption are recorded	2
N-I3	Platform shall maintain a log of the operations done by users	2
N-I4	Log storage shall be tamper-proof	1

Deployment

The deployment of software contains all the steps necessary to make the software available to other users. In the case of the SATO platform, the objective is to make it easier and accessible to others, so that it can be deployed in any infrastructure. This criterion was translated to five non-functional requirements in Table 12.

Table 12: Non-functional requirements from the Deployment criterion

ID	Requirement	Priority
N-J1	The internal components of the platform shall work seamlessly in public and private clouds	2
N-J2	The platform shall be deployed in heterogeneous hardware and software stack	1
N-J3	Components shall explore the isolation and easy deployment provided by virtualisation	1
N-J4	Internal components shall be deployed outside the buildings	1
N-J5	Buildings shall contain only devices and gateways	2

Documentation

Given the intended extensibility of the SATO platform, a good documentation should be provided, in a way that its interfaces and services can be easily used by future integrated platforms or applications. This criterion composes two non-functional requirements that are presented in Table 13.

Table 13: Non-functional requirements from the Documentation criterion

ID	Requirement	Priority
N-K1	Services shall provide well documented APIs	1
N-K2	Components shall be well documented for enabling modularity	2

Scalability

Given the expected scale of the pilot demonstrations in the SATO project (Section 2) and the natural large scale of IoT environments, the SATO platform must support these variable scenarios without becoming a bottleneck for the assessment or the optimisations. Additionally, the platform must deal with changes in the number of devices and adapt the needed resources to support the current workload. This criterion resulted in the four non-functional requirements present in Table 14.

Table 14: Non-functional requirements from the Elasticity/Scalability criterion

ID	Requirement	Priority
N-L1	Services shall support managing dozens of thousands of devices from dozens of buildings	1
N-L2	The platform shall ingest data from dozens of thousands of devices	1
N-L3	The platform shall scale out (and in) to adapt the demand	2
N-L4	Services shall scale up (and down) to adapt the demand.	2

Modularity
Modularity provides means for the SATO platform to support a better decoupling of its functionalities by using modules, where they can be replaced while still maintaining the platform operational. It is also associated with the interoperability, flexibility, or even extensibility since a modular approach, if correctly employed, enables the exchange of different components or integrating new ones. This criterion resulted in the six non-functional requirements from Table 15.

Table 15: Non-functional requirements from the Modularity criterion

ID	Requirement	Priority
N-M1	Components shall be modularised to be replaced if needed	2
N-M2	The platform shall provide a device management service	1
N-M3	The platform shall provide a location service	1
N-M4	Platform shall enable dynamic discovery of components and their features	2
N-M5	Components should be reusable	2
N-M6	Components shall be highly configurable	2

Extensibility
Aligned with the previous criterion, extensibility enables the SATO platform to integrate different functionalities and external platforms to support a bigger set of devices and use cases. This extension must be easy to accomplish in the platform, simply by creating new connectors or separate modules. This criterion results in two non-functional requirements, which are presented in

Table 16.

Table 16: Non-functional requirements from the Extensibility criterion

ID	Requirement	Priority
N-N1	Components shall support extensible features	2
N-N2	Communication shall support extending connectors	1

Flexibility
The SATO platform must be future-prone in a way that unforeseen use cases and scenarios can easily be achieved with little modifications on the platform itself. It means that the designed components must be flexible enough to support unforeseen assessments and optimisations. This requirement was translated into the two non-functional requirements present in Table 17.

Table 17: Non-functional requirements from the Flexibility criterion

ID	Requirement	Priority
N-O1	Components shall support optimisations or the replacement for better modules	2
N-O2	The platform shall be future-prone by supporting generic workflows	2

Open Source
Using open-source tools and open standards is a top priority in any research project that intends to contribute back to the ecosystem and maintain its costs as low as possible. Additionally, using them prevents the platform from being locked in by a specific vendor or provider that may arbitrarily decide to increase its pricing (e.g., Azure, AWS, or Google Cloud). By using open standards, also makes the platform more accessible and easier to integrate with its users while ensuring minimal quality standards. All these aspects are summarised in the three non-functional requirements from Table 18.

Table 18: Non-functional requirements from the Open-Source criterion

ID	Requirement	Priority
N-P1	Components shall favour using of open and standard integration frameworks and communication protocols	1
N-P2	The platform shall support well known communication protocols	1
N-P3	Components shall favour using existent open-source solutions rather than developing new ones from scratch	2

Performance
The performance of the platform is seen as a major goal of the project since we want to have short response times, high throughput, and high availability. This will ensure that the SATO platform can

handle the workload without becoming a bottleneck. In Table 19, we can see this criterion translated into seven non-functional requirements.

Table 19: Non-functional requirements from the Performance criterion

ID	Requirement	Priority
N-Q1	Processing shall occur in useful time	1
N-Q2	The platform shall support resource prioritization	1
N-Q3	Control data (e.g., actuating or adding a new device) shall have high priority	1
N-Q4	Data from the devices (e.g., measurements) shall have low priority	1
N-Q5	Data from devices shall present multiple internal priority levels	2
N-Q6	Processing shall consider data locality	2
N-Q7	Processing shall favour function shipping rather than data shipping	2

Quality

Data quality is an important aspect to enable the proper self-assessment and optimisations that the SATO project proposes to achieve. The platform must collect the inbound data, protect its integrity, annotate it with metadata, and enhance it with standard data models and semantics. Additionally, the platform must detect missing or incorrect measurements and annotate or correct them as the data flows. Table 20 describes the seven identified non-functional requirements resulting from this criterion.

Table 20: Non-functional requirements from the Quality criterion

ID	Requirement	Priority
N-R1	Data shall be enhanced with metadata	2
N-R2	Quality control shall ensure minimal data quality levels for processing	1
N-R3	Missing events shall be detected and inferred from models based on previous measurements	2
N-R4	Incorrect measurements shall be detected, annotated, and corrected	1
N-R5	Events shall be timestamped by the platform	2
N-R6	Location reporting should be supported	1
N-R7	The platform shall be able to discover devices or people close to a location	2

Recoverability
Faulty components decrease the fault tolerance from the system since it may not be able to tolerate future faults. Recoverability is a means of restarting the fault tolerance of the system and of enabling it to achieve its best performance while remaining available. This criterion also applies for lost data, that should be recovered from replicas or backups available in different repositories. Table 21 presents the five identified non-functional requirements associated with this criterion.

Table 21: Non-functional requirements from the Recoverability criterion

ID	Requirement	Priority
N-S1	Faulty components shall be recovered to reset the fault tolerance of the platform	1
N-S2	Snapshot images shall enable the fast recovery of faulty components	2
N-S3	When applicable, passive/backup replicas shall assume the service while the primary replicas are recovered	2
N-S4	When possible, proactive recovery may be employed	3
N-S5	Persistent storage shall be replicated or have backups	1

Efficiency
Given the scale of the foreseen use cases and pilots, the amount of data flowing through the platform can become huge and difficult to handle in useful time. SATO platform must be aware of this risk and must promote the use of efficient procedures and models to reduce the chances of becoming a bottleneck in the project. The four non-functional requirements in Table 22 were identified.

Table 22: Non-functional requirements from the Efficiency criterion

ID	Requirement	Priority
N-T1	The platform shall support event-based communication	1
N-T2	Processing shall have shortcut access to all locations where data is stored	2
N-T3	Processing information shall be optimised with respect to cost functions (e.g., time, space, energy)	2
N-T4	Communications shall consider efficient serialization	1

Economy
Any platform that focuses in optimising a scenario to reduce its costs must require a lower cost than the ones it incurs. Naturally, the SATO platform is no different, and its costs must be considerably smaller than the benefits it will provide through the assessment and optimisation. The final two non-functional requirements identified in this analysis are associated with the economic criterion and are presented in Table 23.

Table 23: Non-functional requirements from the Economy criterion

ID	Requirement	Priority
N-U1	The platform shall enable a cost reduction with SA&O capabilities	1
N-U2	The overall cost of the platform shall be considerably smaller than the economic benefits it brings	1

5. Final Remarks

This deliverable identified the main functional and non-functional requirements for the SATO platform. It identified and described six main functional requirements and more than a hundred non-functional ones (divided into twenty-one non-functional criteria). The presented requirement analysis was based on scenarios from the use cases and pilots foreseen in the SATO project proposal and on the best practices adopted by large-scale interoperable systems.

This requirement analysis is extensive, but it does not prevent from new requirements being identified in the subsequent design and development processes during the SATO project. Notwithstanding, the identified requirements are comprehensive enough to serve as a stable reference to guide the design and implementation of the SATO platform.

Bibliography

[1] A. C. Menezes, A. Cripps, D. Bouchlaghem and R. Buswell, “Predicted vs. actual energy performance of non-domestic buildings: Using post-occupancy evaluation data to reduce the performance gap,” *Applied Energy,* vol. 97, pp. 355-364, 2012.

[2] B. Bordass, R. Cohen, M. Standeven and A. Leaman, “Assessing building performance in use 3: energy performance of the Probe buildings,” *Building Research & Information,* vol. 29, pp. 114-128, 2001.

[3] SRI Consortium, “Project summary 2nd technical support study,” 2018. [Online]. Available: https://smartreadinessindicator.eu/.

[4] Department of Defense, “Systems Engineering Fundamentals,” 2001.

[5] R. B. Grady, Practical Software Metrics for Project Management and Process Improvement, Prentice-Hall, Inc., 1992.

[6] “NIFO - National Interoperability Framewok Observatory,” 19 February 2021. [Online]. Available: https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/glossary/term/european-interoperability-framework.

[7] N. F. N. a. D. L. McGuinness, “Ontology Development,” 12 2 2021. [Online]. Available: https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

[8] J. H. Dean Allemang, Semantic Web forthe Working Ontologist, Morgan Kaufmann, 2008, p. 330.

[9] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi and K. Taylor, “IoT-Lite Ontology,” 26 November 2015. [Online]. Available: https://www.w3.org/Submission/iot-lite/.

[10] L. Daniele, F. den Hartog and R. Roes, “Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology,” in *Formal Ontologies Meet Industry*, Springer International Publishing, 2015, pp. 100-112.

[11] L. Daniele, M. Solanki, F. den Hartog and J. Roes, “Interoperability for Smart Appliances in the IoT World,” in *International Semantic Web Conference*, 2016.

[12] “SAREF: Smart Applications REFerence Ontology,” 2020. [Online]. Available: https://saref.etsi.org/core/v3.1.1/.

[13] "SAREF4ENER: an extension of SAREF for the energy domain created in collaboration with Energy@Home and EEBus associations," SAREF , 13 12 2016. [Online]. Available: https://saref.etsi.org/saref4ener/v1.1.2/. [Accessed 25 2 2021].

[14] "SAREF4BLDG," 2020. [Online]. Available: https://saref.etsi.org/saref4bldg/v1.1.2/.

[15] M. Poveda-Villalón and R. Garvía-Castro, "Extending the SAREF ontology for building devices and topology," 2018.

[16] "Connectivity and intelligence at the edge of IoT," BOSH, 14 12 2020. [Online]. Available: https://developer.bosch-iot-suite.com/service/gateway-software/. [Accessed 25 2 2021].

[17] "EEBus System Architecture," 2020. [Online]. Available: https://www.eebus.org/technology/#system_architecture.

[18] EEBus Initiative e.V., "EEBus Technical Specification Smart Home IP," Cologne, 2019.

[19] G. P. Maren Fiege, "Meet EEBUS, EV charging and more," Supergen Energy Networks Hub, 2020. [Online]. Available: https://www.youtube.com/watch?v=GJXMs3wSpOg. [Accessed 25 2 2021].

[20] EEBus Initiative e.V., "EEBus SPINE Technical Specification Protocol Specification," Cologne, 2018.

[21] "What is oBIX," [Online]. Available: http://www.obix.org/what.htm.

[22] L. Berbakov, N. Tomašević and M. Batić , "D1.3 RESPOND strategy to support interoperability," 2018.

[23] "Common Information Model," 2021. [Online]. Available: https://www.dmtf.org/standards/cim.

[24] "CIM Operations over HTTP," 29 7 2009. [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP0200_1.3.1.pdf.

[25] "buildingSMART," 10 2020. [Online]. Available: https://www.buildingsmart.org/. [Accessed 25 2 2021].

[26] "glTF™ (GL Transmission Format)," 3 12 2020. [Online]. Available: https://www.khronos.org/gltf/. [Accessed 25 2 2021].

[27] "FIWARE Catalogue," 2021. [Online]. Available: https://www.fiware.org/developers/catalogue/.

[28] "Build your own IoT platform with FIWARE enablers," 27 March 2015. [Online]. Available: https://www.fiware.org/2015/03/27/build-your-own-iot-platform-with-fiware-enablers/.

[29] "OGEMA Introduction of Concepts, Terminology and Framework Services," 2021. [Online].

[30] "OGEMA," 2021. [Online]. Available: https://www.ogema.org/.

[31] "EDP RE:DY," n.d.. [Online]. Available: https://www.edp.pt/particulares/apoio-cliente/perguntas-frequentes/pt/equipamentos-e-manutencao/edp-redy/.

[32] "Data-driven optimization through continuous analysis," [Online]. Available: https://new.siemens.com/global/en/products/buildings/energy-sustainability/total-energy-management/analyze-to-optimize.html.

[33] "Desigo CC," [Online]. Available: https://new.siemens.com/global/en/products/buildings/automation/desigo/building-management/desigo-cc.html.

[34] N. Garg, Learning Apache Kafka, Packt Publishing; 2nd Revised edition, 2015.

[35] "Pulsar Overview," n.d.. [Online]. Available: https://pulsar.apache.org/docs/en/concepts-overview/.

[36] "Managing Brokers," n.d.. [Online]. Available: https://pulsar.apache.org/docs/en/admin-api-brokers/.

[37] "Introduction," 2020. [Online]. Available: https://docs.nats.io/.

[38] "What can RabbitMQ do for you?," n.d.. [Online]. Available: https://www.rabbitmq.com/features.html.

[39] "Mosquitto," n.d.. [Online]. Available: https://mosquitto.org/.

[40] "What is a Container?," n.d.. [Online]. Available: https://www.docker.com/resources/what-container.

[41] "What's the Diff: VMs vs Containers," 28 6 2018. [Online]. Available: https://www.backblaze.com/blog/vm-vs-containers/.

[42] "Containers at Google," n.d.. [Online]. Available: https://cloud.google.com/containers.

[43] "What is Kubernetes?," n.d.. [Online]. Available: https://www.redhat.com/en/topics/containers/what-is-kubernetes.

[44] "oBIX Version 1.1," 11 July 2013. [Online]. Available: http://docs.oasis-open.org/obix/obix/v1.1/csprd01/obix-v1.1-csprd01.html.

[45] "SAREF4ENER," 2020. [Online]. Available: https://saref.etsi.org/saref4ener/v1.1.2/.

[46] "Kafka," n.d.. [Online]. Available: https://kafka.apache.org/.

[47] "Apache Kafka," n.d.. [Online]. Available: https://kafka.apache.org/protocol#protocol_partitioning.

[48] EEBus Initiative e.V., "EEBus SPINE Technical Report Introduction," Cologne, 2018.

[49] EEBus Initiative e.V., "EEBus SPINE Technical Specification Resource Specification," Cologne, 2018.

[50] EEBus Initiative e.V., "EEBus System Architecture," 2020. [Online]. Available: https://www.eebus.org/technology/#system_architecture.

[51] "Medium," 13 3 2013. [Online]. Available: https://medium.com/@stephane.maarek/introduction-to-apache-kafka-security-c8951d410adf.

[52] "Streams Security," n.d.. [Online]. Available: https://kafka.apache.org/10/documentation/streams/developer-guide/security.html.

[53] M. Carter, "Apache Kafka Architecture: A Complete Guide," 16 6 2020. [Online]. Available: https://www.instaclustr.com/apache-kafka-architecture/#.

[54] "Quais são as componentes do edp re:dy?," n.d.. [Online]. Available: https://www.edp.pt/particulares/apoio-cliente/perguntas-frequentes/pt/equipamentos-e-manutencao/edp-redy/quais-sao-as-componentes-do-edp-redy/faq-4747/.