IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 13, 2018, accepted August 22, 2018, date of publication September 17, 2018, date of current version October 8, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2867452*

# Experimentation as a Service Over Semantically Interoperable Internet of Things Testbeds

**JORGE LANZA[1], LUIS SÁNCHEZ[1], JUAN RAMÓN SANTANA[1], RACHIT AGARWAL[2], NIKOLAOS KEFALAKIS[3], PAUL GRACE[4], TAREK ELSALEH[5], MENGXUAN ZHAO[6], ELIAS TRAGOS[7], HUNG NGUYEN[7], FLAVIO CIRILLO[8,9], RONALD STEINKE[10], AND JOHN SOLDATOS[3]**

[1]Network Planning and Mobile Communications Laboratory, Universidad de Cantabria, Edificio Ingeniería de Telecomunicación, 39005 Santander, Spain
[2]MiMove Team, Inria, 75589 Paris Cedex 12, France
[3]Athens Information Technology, 15125 Marousi, Greece
[4]IT Innovation Centre, University of Southampton, Southampton SO16 7NS, U.K.
[5]Institute for Communication Systems, University of Surrey, Guildford GU2 7XH, U.K.
[6]Easy Global Market, Espace Beethoven, 06560 Valbonne, France
[7]Insight Centre for Data Analytics, NUI Galway, Galway, H91AEX4 Ireland
[8]NEC Laboratories Europe, 69115 Heidelberg, Germany
[9]University of Naples "Federico II", Corso Umberto I, 40, 80138 Napoli NA, Italy
[10]Fraunhofer Institute for Open Communication Systems FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany

Corresponding author: Luis Sánchez (lsanchez@tlmat.unican.es)

This work was supported in part by the European Project "Federated Interoperable Semantic IoT/Cloud Testbeds and Applications (FIESTA-IoT)" from the European Union's Horizon 2020 Programme under Grant CNECT-ICT-643943 and in part by the Spanish Government by means of the Project ADVICE "Dynamic Provisioning of Connectivity in High Density 5G Wireless Scenarios" under Grant TEC2015-71329-C2-1-R.

**ABSTRACT** Infrastructures enabling experimental assessment of Internet of Things (IoT) solutions are scarce. Moreover, such infrastructures are typically bound to a specific application domain, thus, not facilitating the testing of solutions with a horizontal approach. This paper presents a platform that supports Experimentation as s Service (EaaS) over a federation of IoT testbeds. This platform brings two major advances. First, it leverages semantic web technologies to enable interoperability so that testbed agnostic access to the underlying facilities is allowed. Second, a set of tools ease both the experimentation workflow and the federation of other IoT deployments, independently of their domain of interest. Apart from the platform specification, this paper presents how this design has been actually instantiated into a cloud-based EaaS platform that has been used for supporting a wide variety of novel experiments targeting different research and innovation challenges. In this respect, this paper summarizes some of the experiences from these experiments and the key performance metrics that this instance of the platform has exhibited during the experimentation.

**INDEX TERMS** Experimentation, Internet of Things, interoperability, semantics, testbeds.

## I. INTRODUCTION

Experimentation is one of the basis for technological advances [1]. Being able to test and assess the behaviour and the performance of any piece of technology (i.e. protocol, algorithm, application, service, etc.) under real-world circumstances is of utmost importance to increase the acceptance and reduce the time to market of these innovative developments.

The Internet of Things (IoT) is unanimously identified as one of the main technology enablers for the development of future intelligent environments. It is driving the digital transformation of many different domains (e.g. mobility, environment, industry, healthcare, etc.) of our everyday life. This is happening by realizing the paradigm of more instrumented, interconnected and intelligent scenarios. Instrumented through low-cost smart sensors and mobile devices that turn the workings of the physical world into massive amounts of data points that can be measured. Interconnected so that different parts of a core system, like networks, applications and data centres, are joined and "speak" to each other, turning data into information. And intelligent with information being transformed into real-time actionable insights at massive scale through the application of advanced analytics.

The IoT concept has attracted a lot of attention from the research and innovation community for a number of

years already [2]–[5]. One of the key drivers for this hype towards the IoT is its applicability to a plethora of different application domains [6], like smart cities [7], [8], e-health [9], [10], smart-environment [11], smart-home [12] or Industry 4.0 [13].

Despite the advances that have been accomplished, there is still enormous scope to develop novel and innovative IoT-based solutions that aim at transforming our everyday life. In this respect, real-life experimentation should play a major role in these developments. Interestingly, there are initiatives that, in order to improve these solutions' maturation and significant rollout, try to support the evaluation of IoT solutions under realistic conditions in real world experimental deployments [14], [15]. However, still they tend to lack the necessary scale or they fail to fulfil some key indicators [14], [16]. Nonetheless, large-scale infrastructures enabling the assessment of developed solutions under real-world circumstances are scarce and are not always available for those willing to test their innovations. Moreover, such infrastructures are typically bound to a specific application domain, thus, not facilitating the testing of solutions with a horizontal approach (i.e. fulfilling requirements from different application domains).

Thus, it is deemed necessary to set-up IoT experimentation infrastructures that have the appropriate scale, experimentation realism, heterogeneity, interoperability and openness to facilitate the development of innovative solutions that can actually realize this paradigm of instrumented, interconnected and intelligent scenarios.

This paper presents a platform that has been implemented for enabling Experimentation as a Service (EaaS) over multiple IoT testbeds. In this sense, the key advance with respect to the state of the art brought by the IoT EaaS Platform, which this paper is describing, is twofold. On the one hand, the tools and services underpinning the EaaS concept across federated IoT data sources that reduce the effort to build and run experiments. Experimenters assessing their research on top of this IoT Platform are able to get data from any of the underlying testbeds using a unique set of tools and Application Programming Interfaces (APIs). On the other hand, the interfaces and models provide IoT testbed semantic alignment and interoperability so that the resulting platform increases scale, heterogeneity, experimentation realism and cross-domain innovation. These testbeds federate through the semantic web platform allowing the interoperability and seamless testbed-agnostic access to the services and data that they provide. The paper describes the overall system architecture as well as the tools implemented to support the EaaS paradigm.

In addition to the description of the design principles and the specification of the different building blocks, another contribution of the paper is the validation and evaluation of the instantiation of the platform design that has been created within the H2020 FIESTA-IoT project.[1] The implementation of this instance of the proposed interoperable IoT EaaS Platform does not only imply the development of the different components integrated within, but also the specification of the semantic information models (i.e. ontologies and taxonomies [17]) that constitutes the baseline of the semantic web-based solutions adopted. This evaluation and validation is based on the performance of the implemented instance during the realization of several experiments on top of it. Additionally, it is also based on the feedback received from the experimenters that actually run the abovementioned experiments.

The structure of the remaining of the paper is as follows. Section II briefly reviews existing infrastructures supporting the EaaS concept (with emphasis on IoT-related ones). The integrated IoT EaaS Platform low-level architecture and the different components that are part of it are presented in Section III. In Section IV the workflow for federating IoT testbeds and making their resources available through the Platform is described. Section V summarizes the tools and procedures that experimenters have at hand to access the underlying testbeds datasets in a testbed-agnostic manner and, thus, carry out their experiments. In Section VI, the results of the validation and evaluation of the Platform instantiation are presented. Finally, Section VII contains some concluding remarks and discussion on the scenarios enabled.

## II. RELATED WORK

### A. EXPERIMENTAL INFRASTRUCTURES

The need for IoT experimentation facilities is driven by the effort and expense required to create realistic environments to test new IoT technologies. This has led to the creation of experimental testbeds. Wisebed [18], FIT IoT-Lab [19], Fed4FIRE [20], and GENI [21] are all testbeds that support wireless sensor network experimentation allowing the testing of new communication and application protocols that underpin the IoT domain (in particular looking at improving the properties of reliability, power consumption, performance, etc. in IoT networking environments). Such environments are technology specific and do not support experimentation of new IoT applications and services. In response, SmartSantander [15] provides a large-scale, geographically distributed range of real-world sensors to test new innovative IoT services; LiveLab [22] offers a facility to evaluate human-usage of the technologies; and [23] presents a Mobile Sensing testbed of smart phones to support field-testing of new crowdsourcing applications. While enormously useful in their own right, these higher-layer testbeds are either domain specific (a particular type of experiment or technology domain) or do not consider key IoT development concerns– namely achieving interoperability across domain silos and heterogeneous technologies. The IoT EaaS Platform proposed in this paper is technology and domain agnostic (federating multiple smart city, smart home, crowd-sensing testbeds) to allow experiments that demonstrate IoT interoperability across highly heterogeneous IoT environments.

[1] http://fiesta-iot.eu/

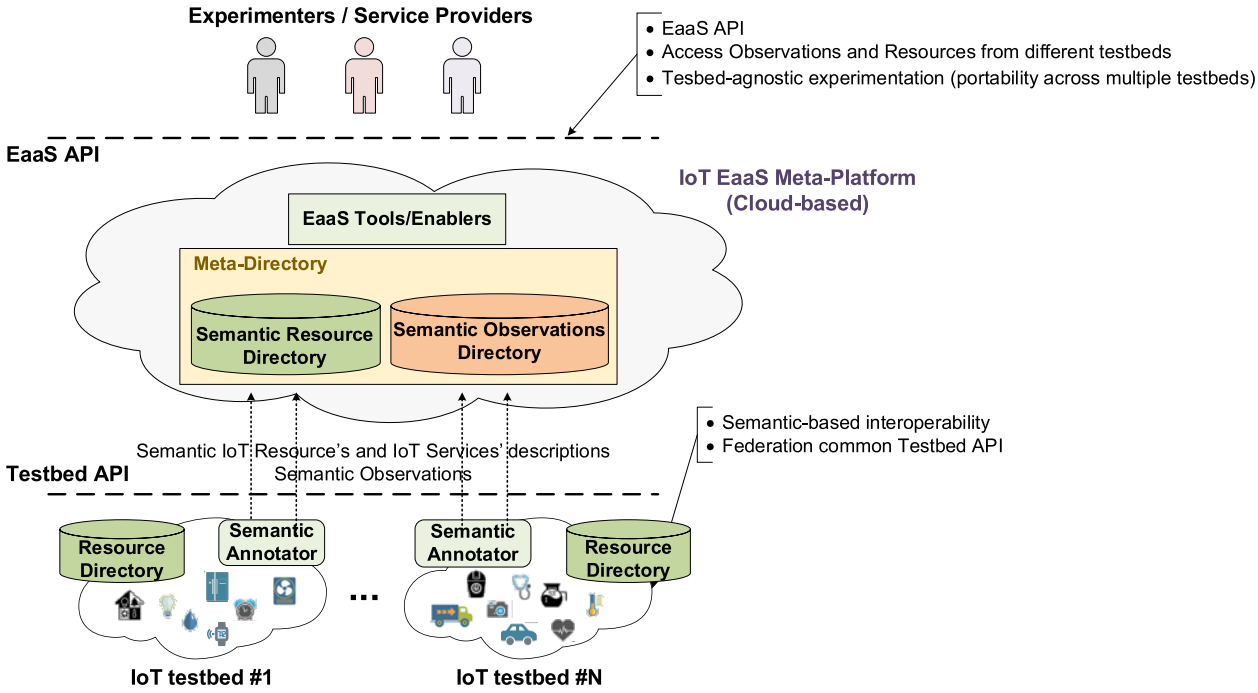


**FIGURE 1.** Abstract IoT EaaS platform and testbed federation concepts overview.

### B. SEMANTIC INTEROPERABILITY

The use of semantic web technologies to query and manage information within federated cyber-infrastructures [24], [25] is being explored as a promising approach to support the necessary coherence among heterogeneous experimental infrastructures. However, most of them make a top-down approach defining only the framework and assessing the meta-directory service using their own ontologies [26], [27], or extensions of established ontologies such as the W3C Semantic Sensor Network (SSN) ontology [28]. They do not take into account the necessities from already deployed infrastructures, and neither define the procedures for them to join their federations. Moreover, some of them are still only design proposals [29] that have not been implemented nor assessed. Finally, those that present some kind of assessment of their solutions' implementation, while supporting the potential of the solution, exhibit a lack of exposure to real-life situations and actual heterogeneous testbeds, including large-scale IoT experimental infrastructures, which would show the true scalability and flexibility of the solutions. At the time of writing, the FIESTA-IoT Platform has already integrated eight different testbeds from heterogeneous application domains (e.g. smart cities, maritime, smart building, crowdsensing, smart grid, etc.) with over 5,000 IoT devices overall which produce millions of observations per day.

## III. FIESTA-IoT PLATFORM KEY CONSIDERATIONS AND ARCHITECTURE

### A. KEY DESIGN CONSIDERATIONS

The main aim of the Platform described in this paper is to enable an EaaS paradigm for IoT experiments. However, instead of deploying yet another physical IoT infrastructure it enables experimenters to use a single EaaS API for executing experiments over multiple existing IoT testbeds that are federated in a testbed agnostic way. Testbed agnostic implies in this case the ability to expose a single testbed that virtualizes the access to the underlying physical IoT testbeds. Experimenters learn once and accordingly use the EaaS API to access data from any of the underlying testbeds.

To this end, the testbeds that aim to participate in the federation have to implement common standardized semantics and the interfaces that have been defined. This enables the meta-platform to access the data that their devices produce as well as the descriptions of their devices and the services that these devices might expose.

As it can be seen in Fig. 1, the central component of the IoT EaaS meta-platform is a directory service (so-called meta-directory), where sensors and IoT resources from multiple testbeds are registered. In the same way, the observations produced by these resources are also stored. This directory enables the dynamic discovery and use of IoT resources (e.g., sensors, actuators, services, etc.) from all the interconnected testbeds.

The key concept behind the federation of IoT testbeds is the specification of a common Testbed API that defines the interfaces to carry out the registration of the testbed resources as well as pushing of the observations to the meta-platform. Besides the actual technologies used for implementing these interfaces, the main feature that underlies the Testbed API is the fact that the information is exchanged in a semantically annotated format.

In this sense, the first main design decision is the use of semantic technologies to support the interoperability between heterogeneous IoT platforms and testbeds. Using a common ontology makes it possible to seamlessly deal with data from different sources. Federated testbeds have to implement their own Semantic Annotators to transform the data they handle internally to the common semantic ontology defined by FIESTA-IoT. Different RDF representation formats (e.g. RDF/XML, JSON-LD, Turtle, etc.) are supported as long as the common ontology is used.

The second major design decision is to take as reference the IoT ARM as defined in the IoT-A project [30]. This decision has brought out, within the IoT EaaS Platform context, the need to comply with the Domain and Information Models defined in the ARM. Thus, the architecture focus on defining a canonical set of concepts which all IoT platforms, which can be part of the federated IoT EaaS Platform can easily adopt. The adoption of these essential concepts only require from underlaying testbeds a straightforward tuning of the models that they handle internally. In this sense, independently of which internal model the testbeds uses, whether it is proprietary or based on existing standards [31], [32], they should be able to find in a straightforward manner how to map the internal modelling to the canonical concepts managed within the IoT-related ontology used as a basis for the Platform. The aforementioned tuning of models basically consist on mapping the internal structure of information to the one that uses the ontology as a basis. The less number of concepts to map and the more fundamental these concepts are, the less the chances to have existing IoT platforms that are unable to perform the mapping between their internal data model and the IoT-related ontology that is employed to enable interoperability among the federated IoT infrastructures.

The foremost aspect that these choices imply is that the ontology that is used to regulate the semantic annotation of the testbeds' resources is only bound by the core concepts that compose the aforementioned ARM Domain and Information Models. These core concepts are:

- A Resource is a "Computational element that gives access to information about or actuation capabilities on a Physical Entity" [30].
- An IoT service is a "Software component enabling interaction with IoT resources through a well-defined interface." [30].

These concepts conform the baseline for representing the devices and overall IoT infrastructure. However, there is still a major concept that is not tackled within the ARM models. This concept relates to the actual data that is gathered by the devices and offered through the services that expose them. It is the Observation concept:

- An Observation is a "*piece of information obtained after a sensing method has been used to estimate or calculate a value of a physical property related to a Physical Entity*".[2]

[2]Observation description from Semantic Sensor Network (SSN) Ontology. https://www.w3.org/2005/Incubator/ssn/ssnx/ssn#Observation

The fact that the IoT EaaS Platform is not bound to any ontology makes it design fundamentally re-usable and extendable.

### B. FIESTA-IoT PLATFORM FUNCTIONAL ARCHITECTURE

The IoT EaaS Platform has been designed and implemented having all these considerations in mind, both in terms of enabling EaaS and allowing IoT testbeds federation. Fig. 2 shows the Platform functional architecture.

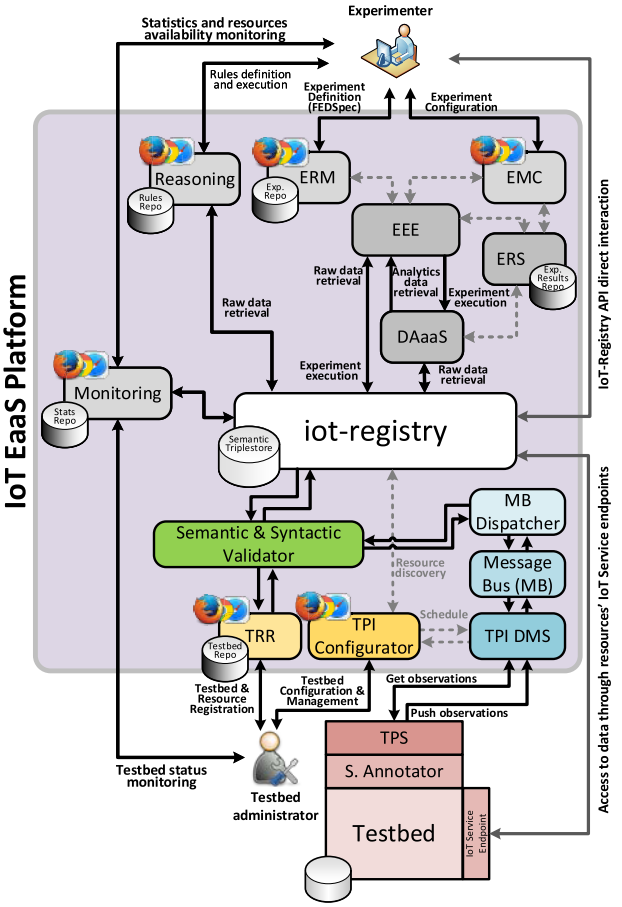


**FIGURE 2.** IoT EaaS platform detailed functional architecture.

At the core of this architecture, the IoT-Registry is its key component. It stores all the semantic information related to underlying testbeds IoT devices and the observations that they generate. Moreover, it exposes the interfaces necessary to access this data.

On top of the IoT-Registry, the architecture consists of the additional set of tools and APIs underpinning the EaaS concept. They allow experimenters assessing their research on top of the IoT EaaS Platform to get data in a testbed-agnostic manner. Moreover, several experiment management components, namely Experiment Execution Engine (EEE), Experiment Registry Module (ERM), Experiment Management Console (EMC) and Experiment Result Storage (ERS) ease the experiment creation and management and the result extraction. They allow the experimenter to define an experiment in an XML-based document, schedule its execution in

an unattended manner and store the results for later retrieval. ERM and EMC are available via the Platform Portal, a web-based UI open to the experimenters and testbed providers (represented in Fig. 2 with browser icons).

In addition to these components, added-value services are also provided as part of the Platform experimentation tools portfolio. Through the Analytics and Reasoning modules experimenters can easily get added-value data without the need to implement the algorithms themselves. These modules can be used within the workflow of the experiment to get the raw data from the IoT-Registry and generate already processed data that fits with the experimenter needs.

Below the IoT-Registry the IoT EaaS Platform is focused on the interfaces and models supporting IoT testbed semantic alignment and interoperability so that the resulting platform has increased scale, heterogeneity, experimentation realism and cross-domain innovation. Still at the testbed side, two components must be implemented. The Semantic Annotator and the Testbed Provider Services (TPS) respectively transforms the data model used internally at the testbed into semantically annotated data (based on the IoT-related ontology defined for the specific instance of the IoT EaaS Platform) and exposes the interfaces for the Platform to access that data. The TPS interacts with the Data Management Services (DMS) already at the Platform side of the architecture. The DMS proxies the observations that arrives at the IoT EaaS Platform towards the IoT-Registry, where they are stored indefinitely. Since the Platform does not only manage information related to observations generated by underlying testbeds only but also the descriptions of the actual IoT devices (i.e. sensors, actuators and tags), the Testbed and Resource Registration (TRR) module exposes the necessary interfaces to register the descriptions of the testbeds' resources. This registration is done either via the Platform Portal or through the TRR API. Before any RDF document is stored in the IoT-Registry, its compliance with the IoT-related ontology employed as baseline for interoperability has to be validated. Otherwise, the data inserted into the repository could be flawed and cause issues while querying afterwards. The Semantic Validator is in charge of this assessment both for the observations and the resource descriptions.

Finally, all the interfaces exposed by the IoT EaaS Platform are secured using HTTPS and the corresponding authentication and authorization filters. Every query received has to pass through a Policy Enforcement Point (PEP) which checks if it contains a valid security token and if that token actually belongs to a user authorized to make such query.

### C. IoT-REGISTRY

IoT-Registry's main function is to store all the (semantic) resource descriptions and observations that the underlying testbeds provide. On top of this ''collector'' behaviour, it implements a fully-fledged REST API that allows the interplay between users and the stored information. Fig. 3 shows the internal architecture of the IoT-Registry.

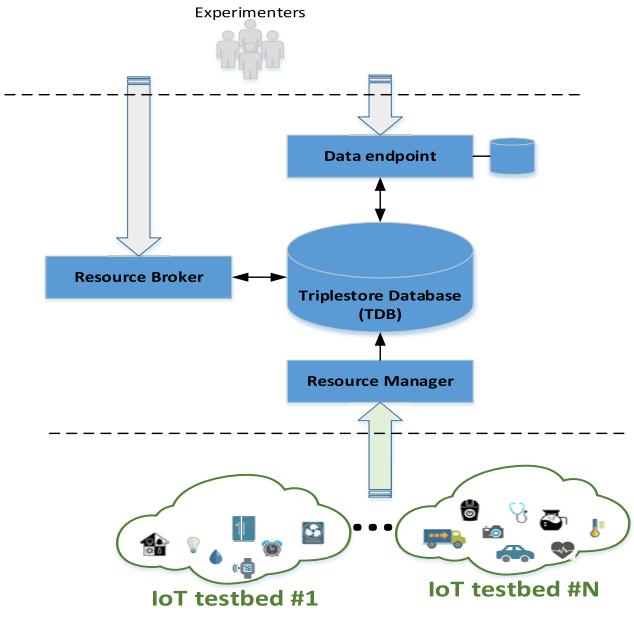


**FIGURE 3.** IoT-Registry internal architecture.

At the core of the IoT-Registry is the Triplestore Database (TDB) that provides the storage capacity for aggregating the Resource Descriptions from the devices belonging to the federated testbeds as well as the Observations that these devices are constantly producing. In this sense, the TDB internal structure follows the canonical concepts defined by the ARM information model. By using these concepts as a basis for its internal structure, it is able to adapt to different IoT-related ontologies.

However, this is only the storage part of the component and the actual functionality of the module is implemented via other sub-systems. The first of these functional modules is the Data Endpoint that is responsible for exporting the SPARQL endpoint of the TDB's query engine into a web-based API. It mainly acts as a proxy getting the SPARQL queries that are carried in the body of the HTTP requests, injecting them into the native SPARQL endpoint of the TDB and getting back the corresponding response within the HTTP response packet.

The remaining two components, namely the Resource Manager (RM) and the Resource Broker (RB), transform the IoT-Registry from a regular Semantic Datastore into an enabler of the Web of Things paradigm. The key idea is that the services exposing the underlying IoT devices (i.e. sensors and/or actuators) are accessed using a truly web-oriented style. Both, the testbed-agnostic nature of the IoT EaaS platform and the service-oriented character of the IoT ARM [30], which underlies all the Platform architecture, are behind this behaviour. Firstly, the IoT-Registry hides the underlying resources by exporting a homogenized URI under a common domain namespace for each of the federated testbeds. It then provides a brokering mechanism that enables unified and proxied access to the underlying resources and the service endpoints that are used to expose them.

#### 1) STORAGE STRUCTURE OF THE TDB

As a result of the semantic modelling that underlies the design of the Platform, the information that is stored at the IoT-Registry relates to two different, but tightly bound, realms. On the one hand, the descriptions of the resources that form the underlying testbeds and, on the other hand, the observations made by them. The internal structure of the TDB follows a similar approach. The implementation of the Jena-based query engine has two different graphs that are virtually merged into a global one, as can be seen in Fig. 4. The resources and observations graphs store the resource descriptions of the IoT devices and the observations that they generate, respectively.

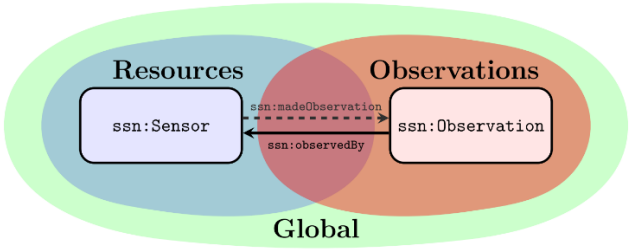


**FIGURE 4.** IoT-Registry TDB internal structure.

The linked graphs that the instances of each of these items form are mostly independent and, indeed, can be queried individually if the experimenter is interested only on information related to one of them. For example, the experimenter can look for the Service that is exposing any of the IoT devices using the typical what (i.e. physical phenomenon observed) and where (i.e. location) discovery criteria. For this search, only the resources graph should be explored by the query engine optimizing this way the discovery and access performance. Similarly, if the experimenter is interested on the data contained on the actual measurements collected by the sensors as they are also self-contained in terms of geo-location, timestamp and phenomenon observed.

However, in the cases where the experimenter is looking for extra information about the IoT device that has produced an observation (e.g. accuracy, sensing procedure or other metadata), this information can only be obtained from the resources graph. If the two graphs weren't virtually bound, the experimenter would have to execute two different queries. One over each of the two graphs. The solution adopted caters for the flexibility of allowing optimized queries when they target only one of the graphs but at the same time allows more complex queries looking for information that is stored on both of them.

#### 2) DATA ENDPOINT

SPARQL is known to be the most common and widely used RDF query language. Therefore, it is sensible to offer a fully-fledged SPARQL interface, as part of the IoT Registry module that enables the support for this kind of semantic queries. The Data Endpoint (DE) module implements this functionally by enabling a direct SPARQL endpoint.

The DE is a conformant SPARQL protocol service as defined in the SPARQL Protocol for RDF (SPROT) [33]. It allows users to query a knowledge base via the SPARQL language. Results are returned in any of the common data representation formats, namely JSON, XML, CSV, etc. The default endpoint runs the query on the "global" graph. However, it is also possible to limit the scope of the query to just the Resources or the Observations graph.

Moreover, it also offers a system for the storage of queries so that its execution can be programmed without having to include the complete SPARQL sentence at every request. This additional functionality would make it easier to share knowledge between experimenters or testbeds and smooth the learning curve when it comes to cope with the specific features of the IoT-related ontology that is employed.

Finally, an additional functionality has been added to the DE so that the stored SPARQL queries can be dynamically adapted and used as templates rather than as static queries. To achieve such a feature, the REST API wrapping the DE allows some variables to be replaced with input parameters in the GET/POST requests based on a set of pre-defined conventions. This feature has been added with a twofold objective. On the one hand, it promotes sharing queries, thus giving rise to a sort of "crowd-sourced" catalogue. Moreover, it enables the creation of optimized queries resolving recurrent demands from experimenters. This way it is possible to create a "best-practices" catalogue open to experimenters. On the other hand, this option reduces the overhead and eases the action of executing multiple times the same SPARQL sentence as caching can be used to enhance the query engine performance.

#### 3) RESOURCE MANAGER

The Resource Manager (RM) exposes the single-entry point for all the testbeds to register their IoT Resources' descriptions. Its main role is to homogenize the descriptions received from the different testbeds. After syntactically checking the annotated descriptions and guaranteeing that they are compliant with the specific IoT-related ontology selected for that instance of the IoT EaaS Platform, the RM transforms the URI for all the resource descriptions in order to make them belong to the common namespace. This process basically consists of overwriting the bindings that points to the original testbeds' domains included in the annotated resource descriptions. These bindings are transformed to the common meta-platform domain so that every entity identifier and/or IoT Service endpoint, independently of which testbed they belong to, are exposed as if they belonged to a unique graph, namely the federation graph. For example, the resulting transformed URI for one of the testbeds original URI:

http://api.smartsantander.eu#SmartSantanderTestbed

becomes:

https://platform.fiesta-iot.eu/iot-registry/api/testbeds/a1yp9GcKEPw37Bx5rslgRI4QLSNCwEwBatCIOe_W0dHZCmzj2WmkExz3qoNuvWg1pueAXn1Li0JrNjvBiQwV3Q==

Therefore, all the semantically annotated descriptions generated by the testbeds are stored in the Triplestore Database following the testbed-agnostic paradigm adopted for the design of the IoT EaaS Platform. Once the necessary adaptations to the resource descriptions have been done and internally recorded for future use by the RB, the RM stores them into the TDB.

While the communication interface between the RM and the TDB is based on semantic requests, the interface with the testbeds is based on standard HTTP encapsulating semantic documents.

#### 4) RESOURCE BROKER

Apart from the extraction of data from the TDB by executing SPARQL queries, the Platform supports the access to the services that directly expose the underlying IoT devices [34].

The Resource Broker is the component in charge of enabling the access to IoT devices' services while keeping the testbed agnostic nature of FIESTA-IoT and homogenizing the way of accessing them for the experimenter.

Graph's nodes URIs are transformed for them to belong to the unified Platform namespace. This transformation makes the service endpoint to target the IoT EaaS Platform namespace and more specifically the IoT-Registry. The RB intercepts the requests made to the transformed URIs and forwards it to the corresponding testbed endpoint. This process is carried out internally at the RB so that for the experimenter it is completely transparent and it gets the service result without having to care about the specific testbed requirements. The RB manages any and all specific requirements (e.g. authentication method, etc.) imposed by each of the underlying testbeds.

### D. EXPERIMENT DEVELOPMENT, DEPLOYMENT AND MANAGEMENT

An experiment is defined as "*a test under controlled conditions that is made to demonstrate a known truth, examine the validity of a hypothesis, or determine the efficacy of something previously untried*".[3] Nevertheless, our EaaS Platform focus on data-oriented experimentation where experimenation can be performed on the stored IoT data. The modules that address all the steps in the execution of an experiment (i.e. development, deployment and management) are shown in Fig. 2.

The core of the experimentation support subsystem is the Experiment Execution Engine (EEE). This module essentially schedules or deploys the experiment based on the provided experiment specifications. The EEE exposes APIs that are broadly divided into 5 categories: scheduling, polling, subscription, monitoring and accounting. Scheduling API enable creating a recurrent job that executes the query included as part of the experiment specification. It also provides information (general description and status) about the created job, API to change the execution status of the job (start, stop, and resume job), change schedule parameters, and API to delete the job. The polling API provides a way to execute the experiment once and not to schedule it. The subscription API let experimenter subscribe or unsubscribe the public experiments. On the other hand, the accounting and monitoring APIs provides log information and status information about the execution of the experiments.

EEE fetches from the Experiment Registry Module (ERM) the information about the experiments that is to be executed. ERM basically stores the experiments' specifications and provides the interfaces to handle the storage process (e.g. saving, deleting, sharing, etc.). EEE is accompanied by an experiment controlling and management user interface (Experiment Management Console or EMC) that enables experimenters to view an execution summary and control the execution of their experiment. Once an experiment is executed by the EEE, the results are sent to experimenters. The experimenters need to enable a Receiver on their side to receive the results. In case the results are not delivered to the experimenter, the results are stored in an Experiment Result Storage (ERS) repository where experimenters can download the results at will.

Dedicated APIs, which can be used by experimenters to develop their own experiment workflow, complement the above tools. In the case where experimenters do not want to use the graphical interfaces of these tools, they can use the APIs of these modules or perform querying directly on IoT-Registry using the public IoT-Registry APIs.

Another set of added-value services (described in Section III.F) are provided to help experimenters with the IoT data stored within IoT EaaS Platform.

### E. TESTBED PROVIDER INTERFACE

The Testbed Provider Interface (TPI) specification considers the main functionalities and properties that should be exposed by IoT testbeds in order to enable their integration within the EaaS Platform for the purposes of testbed-agnostic experimentation. The TPI is a set of RESTful web services whose definition has been driven by various requirements, including flexibility and ease in the integration of testbeds, support of mainstream IoT standards for data and services representation and compatibility with existing IoT testbeds.

[3] A. H. Soukhanov, K. Ellis, and M. Severynse, The American Heritage Dictionary of the English Language. Boston: Houghton Mifflin, 1992.

The TPI spans across two different realms (cf. Fig. 2). The first is the EaaS Platform side with the TPI Configuration & Management layer that controls the functionality of the TPI by utilizing the offered user interface for the User (Testbed provider). The second is the testbed side with the Testbed Provider Services (TPS) API where the Testbed Provider (TP) has to implement a set of services that enables the management and manipulation of the offered data.

A testbed may expose internally various standard and/or proprietary interfaces in order to interact with the sensor data. Thus, a list of core services (TPS) that should be exposed by a testbed in order to enable different connection methods to the EaaS Platform have been specified.

The behaviour of these methods is controlled from a set of services, provided by platform itself (so-called TPI Data Management Services – DMS). They enable the TPs to consume and control the TPS services that their testbeds expose either by identifying a specific schedule or by enabling a data stream connection.

In Fig. 5 we can see a simplified diagram of the different service interaction between the DMS services and the TPS ones for applying the different DMS functionalities described below. These services are grouped into two types according to the relation established between the testbed and the EaaS Platform, namely get-based and push-based. The TP can choose to either control the schedule of when to push the data (TPS Push Observations towards DMS) or let the platform control the schedule (DMS Get Observations from TPS).

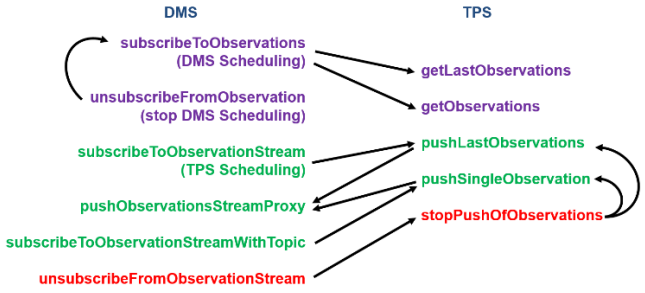


**FIGURE 5.** DMS-TPS service interactions.

In the Push case, the TP triggers the TPS once in order to start pushing. The observations are then sent to the EaaS Platform as they are produced, or based on a scheduled controlled within the testbed itself. In the Get case, the TP specifies a schedule so that the testbed is polled at the configured frequency in order to retrieve the observations.

In order to be able to initiate this configuration and set up process, the TPs need to register first the metadata of their testbeds and resources. This is done by utilizing the services that are exposed by the Testbed & Resource Registration (TRR) (cf. Fig. 2). The TPI Configurator, which is a User Interface component, enables the TP to discover the available resources, and manage the data retrieval process. It utilizes the IoT-Registry, TRR and TPI DMS services for that.

#### 1) TESTBED PROVIDER SERVICES (TPS)

As it has been described, in order to enable the "plugability" of the testbed to the EaaS Platform, it has to implement and expose at least one (get or push) of the TPS services.

For the Get case, the **getLastObservations** and the **getObservations** services responds with the latest observations from a list of sensors, and with the observations from list of sensors for a specific time-period, respectively. The list of sensors from whom the observations are retrieved is the input parameter for both services.

For the Push case, the **pushLastObservations** and the **pushSingleObservation** services correspondingly initiate a stream at the testbed side that pushes the observations from a list of sensors or from a specific sensor towards a specific endpoint at the TPI DMS. Both the list of sensors or the specific sensor from whom the observations must be pushed are the input parameters for each service. The **stopPushOfObservations** service stops the pushing of observations initiated by the said services and must be implemented in combination with them.

#### 2) DATA MANAGEMENT SERVICES (TPI DMS)

Regarding the TPI DMS services, which enable the TP to manage the services exposed by the testbeds' TPS, for the Get case, the **subscribeToObservations** service queries the corresponding get-based TPS service based on a specific execution schedule and pushes the observations in the response to a specific endpoint. The **unsubscribeFromObservation** service stops the periodic polling initiated before.

For the Push case, the **subscribeToObservationStream** service instructs the testbed's TPS to push the observations from a specific sensors' list to a specific endpoint (**pushObservationsStreamProxy**) as soon as they are generated. The said **pushObservationsStreamProxy** service is used in combination with the previous onw. It essentially creates a "proxy" between the TPS and the Message Bus (MB) for the testbeds to push their annotated observations measurements. Alternatively, the **subscribeToObservationStreamWithTopic** service triggers a similar behaviour on the TPS, which, in this case, starts pushing the observations directly to the MB using the identifier of the sensor that produced the observation as queue topic. Finally, the streams initiated by the previous two services are stopped using the **unsubscribeFromObservationStream** service.

### F. ADDED-VALUE SERVICES

#### 1) SEMANTIC ANNOTATION VALIDATION

In order to guarantee the validity and the consistency of the data stored in the IoT-Registry, all the input semantic annotation of resources and observations are validated before the storage.

The validation can be configured to use any ontology as the reference ontology. In the current FIESTA-IoT use case, the FIESTA-IoT ontology is set as the reference ontology [17].

The Semantic and Syntactic Validator (cf. Fig. 2) performs the validation at several levels:

1. ***Lexical check***. It consists of verifying the correctness of RDF serialization regarding to the declared type (e.g. checking the XML format if the annotation is declared to be in XML).
2. ***Syntactic checks***. The syntactic check consists of verifying the correctness of the "syntax" of the RDF triples represented by the underlined serialization format, more specifically:
   a. *Un-typed resources and literals*. Here resource refers to instances of a class, and literal refers to a textual or numerical value. The type of resource or literal is the link of an annotation back to the ontology that enables the semantic capabilities. Any un-typed element presented in an annotation is problematic towards the semantic interoperability.
   b. *Ill-formed URIs*. They are checked against RFC3986[4] that defines the syntax of URI.
   c. *Problematic prefix and namespaces*. Namespaces play the role of linking the annotation to the reference ontologies and vocabularies. A one-to-one mapping between the prefix and namespace is essential and shall be checked to ensure correct referencing.
   d. *Unknown classes and properties*. A prerequisite of semantic interoperability is that all the resources use an agreed vocabulary. As consequence, if any resource uses in its annotation a class or property that is not defined in the reference ontology, other resources would have no way to understand it, so that the semantic interoperability is impossible.
3. ***Semantic checks***. Following a successful syntactic validation, the semantic check consists of verifying the consistence of the semantic annotation regarding to the reference ontology:
   a. *Problematic relationship or inheritance*. Checks whether there is a model of Ontology (i.e. whether there exists a (relational) structure that satisfies all axioms in this ontology.
   b. *Consistency of A with respect to B*: determine if individuals in A do not violate descriptions and axioms described by B.

An annotation is considered "valid" only if all the above aspects are checked without errors. If any error occurs, the annotation is not pushed to the IoT-Registry for storage, and the data-provider receives a response from the validator containing a test report indicating what is wrong in the submitted data. If the annotation is valid, it is pushed to the IoT Registry, and a response containing the URI of the registered annotation in the IoT-Registry is returned.

[4]Uniform Resource Identifier (URI): Generic Syntax. https://tools.ietf.org/html/rfc3986

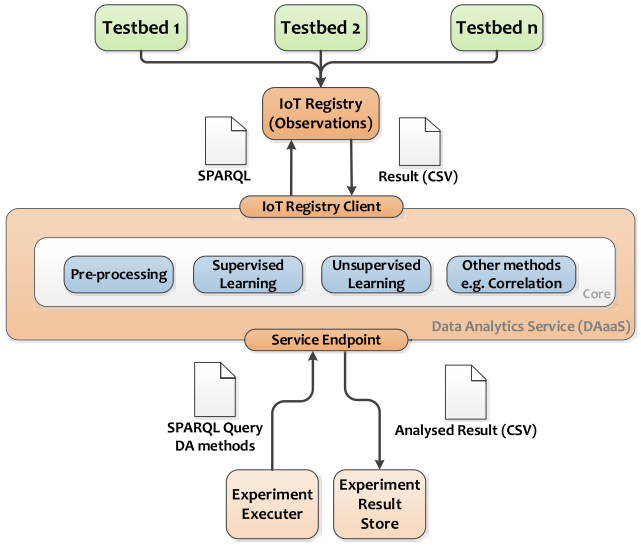


**FIGURE 6.** Analytics service interaction.

### 2) ANALYTICS TOOL

To maximise the added value of the data being extracted from the federated testbeds for the experimenter, it is important to provide data analysis tools. As a result, a Data Analytics web service (DAaaS) based on the Knowledge Acquisition Toolkit (KAT) [35] has been developed for the EaaS Platform to provide open access data analysis tools for data consumers as a web service. Such a tool provides the following benefits: for novice/beginner data consumer, the tools that would enable them to analyse and obtain useful information. While for the more advanced/experienced user providing the most effective tools for a given data set.

The methods implemented as part of the Analytics tool are mainly based on data pre-processing techniques and machine learning techniques. For pre-processing, they involve the removal of corrupted or noisy data points from the original raw time series data. For machine learning, unsupervised machine learning techniques enable the experimenter to discover patterns of interest in the data set being analysed. Supervised learning techniques are included to aid an experimenter either to determine a relationship between a set of input and output data points, or to obtain an estimate of the output data points given the input data points.

There are also other methods that provide spectral analysis and data dependency estimation for the experimenter. Spectral estimation tools are particularly useful for designing digital filters for removing noise, while data dependency estimation tools are particularly useful for linear regression.

As it is shown in Fig. 6, to invoke the Analytics service, a HTTP POST request must be made. The body of the request contains a JSON object that encapsulates the list of methods and the corresponding parameters to be applied, the SPARQL query of which the retuned dataset will be based on, and the SPARQL endpoint where the dataset can be queried and obtained.

### 3) REASONING TOOLS

Apart from their capacity to enable interoperability, the key feature of semantics is enabling the extraction of knowledge out of information. This happens through ''reasoning'' engines that are mainly software components that allow the inference of logical consequences from a set of rules. A key part of the reasoning engines is the set of ''rules''. They are normally specified by the end user (when they are linked with applications) or they are following the ontologies of the system.

In this sense, the EaaS includes a reasoning module that eases the process to extract knowledge out of the measurements generated by the integrated testbeds to the experimenters. The reasoning engine within the module is a rule-based engine that is able to infer logical consequences from the testbed measurements, simplifying the creation of rules. The reasoning engine is developed based on the Apache Jena open source framework.[5] The reasoning module allows the experimenters to define rules in the form of expressions ''if (condition) then (result)'' as below:

- If (temperature) $>$ (25degrees) then (notify_hot)
- If (temperature) $<$ (19degrees) and (humidity) $>$ (60%) then (notify_unhealthy).

The architecture of the reasoning module is shown in Fig. 7. It provides three APIs for creating a rule template, registering a new rule for a sensor or a set of sensors and executing the rule. The rules are stored in a MySQL database. The engine is connected to the IoT-Registry for getting the sensors' descriptions and observations. The end users (experimenters) can access the reasoning module's functionalities either through a simplified graphical interface or through the APIs, which facilitates the way they can integrate the reasoning engine in their applications.

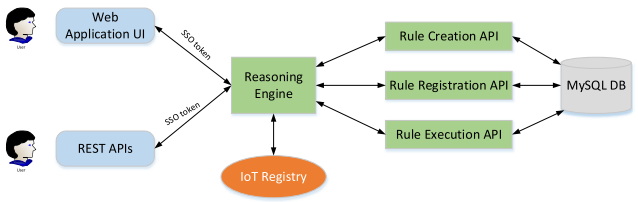


**FIGURE 7.** Reasoning engine architecture.

### 4) ANNOTATOR AS A SERVICE

Data arriving to the EaaS Platform from the testbeds has to be semantically annotated using a reference ontology. Thus, it is necessary to map the data format managed internally by the testbed to RDF documents complying with the selected reference ontology. The Annotator as a Service (AaaS) module lowers the burden for the TPs as they do not have to implement this mapping completely but just take, from their information models, the pieces of information that map into the desired reference ontology's classes. AaaS receives as input a JSON object with that pieces of information (organized using a pre-defined JSON Schema) and generates the corresponding RDF document. This way, the integration effort for the TP is significantly reduced.

### 5) TESTBED AND PLATFORM MONITORING

In order to give a fast overview of the existing data and the overall situation of the resources, the Testbed Monitoring component is integrated. It helps the experimenters to check in advance the situation of testbeds or resources involved in their experiments. It also helps testbed owners to know if their data is still inserted correctly into the Platform. The user of the Testbed Monitoring can see at the overview page the status of the connected testbeds by showing how many sensors have send an observation within the last day and the total number of registered sensors. A detailed view per testbed is available which lists all sensors of the testbed. Per sensor the meta data like unit and the latest observation can be seen. Per sensor a graph of the last observations can be shown.

Besides of these graphical features, the Testbed Monitoring provides an API that serves the data used in the GUI in JSON-format. Moreover, a notification system is provided so that users can configure the module to send a notification to them when the configured threshold is reached. This can be used, for example, to inform testbed owners that no more data is inserted into the platform anymore. Background tasks will analyse the provided data in order to find anomalies in the data streams that could help to find sensors that do not behave correctly.

## G. DATA SECURITY: AUTHENTICATION AND ACCESS CONTROL

The EaaS Platform provides access to IoT data originating from multiple IoT testbed sources (including sensor data that may or may not contain personal information about people). Here, there are a number of challenges that must be addressed in order to create a secure infrastructure, which protects the IoT data resources, the users of the EaaS Platform, and the privacy of any observed persons.

The proposed architecture is secure-by-design and its implementation puts in place access control solutions (using OpenAM[6] security software) at all critical points in the architecture. This is a PEP (Policy Enforcement Point) and PDP (Policy Decision Point) pattern. That is, where IoT data is either requested or published to the Platform – the authorization of the user performing this action is evaluated against defined security policies and the access decision is enforced.

This framework then provides the following key elements of a secure IoT infrastructure:

1. ***Data authentication***: Data retrieved by experimenters must originate from authentic IoT testbed sources. Federated testbeds must authenticate themselves and send

[5]Apache Jena Open Source framework. https://jena.apache.org/index.html

[6]ForgeRock Identity Platform: Access Management. https://www.forgerock.com/platform/access management/

data via a secure encrypted channel. Only authorized IoT sources can publish data to be available to experimenters.

2. ***Experimenter access control***: The Platform controls access to data to experimenters. By default all data is protected to be only available to experimenters. However, IoT data providers (testbeds) can also set up policies to control which experimenters (or groups of experimenters) can access to their data.
3. ***Subject privacy***: observed persons must provide their consent for these observations to be used by experimenters. Where they do not provide consent, then the data is not included (or made available) in the platform. The Platform requires and checks that individual testbeds enforce this policy.

In summary, the whole EaaS process is secured by fine-grained access control that can ensure that data is accessed securely and in line with existing privacy regulations.

## IV. TESTBED FEDERATION

Integrating a testbed within the EaaS Platform can be achieved by completing a set of steps 1) Develop your annotator and TPS; 2) Get certified by the platform owner; 3) Register your testbed and resources; and 4) Configure your resources.

As it has been already described, data arriving to the Platform from the testbeds has to be semantically annotated using the selected reference ontology that is used as a basis for guaranteeing interoperability. So, for the first step TPs can either develop the annotator themselves or use the Annotator as a Service API (cf. Section 3.F.3).

After successfully generating the testbed's annotator the TP should decide on how the captured observations are going to be provided, this means if the ''Get'' or the ''Push'' methodology is going to be used, and develop the TPS accordingly (see Section 3.E above). In order to facilitate the TP with the TPS development a skeleton component implementing all the required services can be easily provided which would only require the testbed's internal data access and annotator integration.

After successfully completing the TPS implementation the next step would be to validate the implemented TPS and annotator. In order to go over this step, the EaaS Platform includes a Certification Portal that can be used by the TP to get certified.

The next step would be to register the available testbeds and resources to the IoT EaaS Platform. The TP can make use of the tools at the Platform Portal UI for this process.

Finally, the TP should instantiate and schedule the data pushing (testbed controls the scheduling) or retrieval (platform controls the scheduling) whether using the TPI configurator tool or directly calling the DMS services (cf. Section 3.E).

## V. EXPERIMENT AS A SERVICE WORKFLOW

In order to utilize the provided experimentation tools, the experimenter has to create an Experiment Description Specification, so-called EDSpec, which serves as a Domain Specific Language (DSL) for the experimentation tools to know which the experimentation workflow to be followed is. EDSpec is an XML document that contains Experiment Model Objects (EMO). An EMO contains the description and domain of interest of the experiment, and Experiment Service Model Objects (ESMOs). ESMO is the main entity that enables EEE to perform experiment related task. Note that we interchangeably call an ESMO as a job when referring to an ESMO in the context of EEE. This is because EEE creates a recurring job based on the specified parameters. Essentially, an ESMO contains a job description (such as scheduling tag parameters, query to execute, tags notifying where the experiment output should be sent, if result set is empty whether to report to the experimenter or not, list of dynamic attribute tags used within the query) for the EEE to schedule and execute it accordingly.

An EDSpec can be created mainly in two ways: (i) using an experiment editor which provides a graphical user interface to ease the process, or (ii) using any XML editor tool. If the EDSpec is created using an experiment editor, it is directly stored in the ERM when the experimenter saves the EDSpec. However, if an XML editor is used, the experimenters are required to store the EDSpec using ERM client. Using ERM client experimenters are able to review existing EDSpecs, save new EDSpecs, and delete existing experiments. In addition to the user interface, the ERM client also exposes an ERM API that can be used to programmatically manage the experimenters' EDSpecs (i.e. as the experiment editor does). Once the experiment is saved using either of the two methods described above, it is essential that the experiment is enabled for the execution using EEE. As described in Section 3.D, the EMC (a client for EEE) is used to perform such an activity. Within EMC, experimenters first need to select a particular experiment object (EMO) whose service (ESMO) they want to enable then use the interface to view information about the ESMO, start/stop the schedule (enable it for execution), view execution log graphs that include run time and data received information views or clear the execution history. In addition to the above functionality, experimenters can subscribe to already existing service models that have been stored and made publicly available within the EaaS Platform. Such feature enables the experimenters to leverage from already defined services. Once subscribed, experimenters can also unsubscribe the experiment using the EMC. EMC internally uses the EEE APIs (cf. Section 3.D) upon request from the experimenters' actions over the graphical user interface available through the Platform Portal.

The execution of the query is managed in two ways according to the given parameters. If within ESMO, the ''widget'' tag is specified the EEE executes analytics API. The analytics API then executes the query in the ESMO in two phases: first,

it executes the query on the IoT-Registry and retrieves the results and then analysing the results based on the attributes set in the widget. On the other hand, if the ''widget'' tag is not specified the query is executed directly on the IoT Registry DE. As the scope of the query can span both the resource and observation graphs, the EEE executes the queries using the global graph (c.f. Fig. 4).

Once the ESMO is executed using either the analytics tool or IoT-Registry directly, it is essential to transfer the obtained result set to the experimenter. If EEE executed the query on the IoT-Registry directly, after obtaining the result set, it sends the results to the experimenters using the endpoint that they have to specify before running their experiments. Note that, this endpoint is provided in the ESMO. The result-set is sent as a multi-part file to enable large results to be transferred successfully. If the sending fails due to any reason (network failure or service location unavailable), the EEE stores the results into the ERS. The experimenters can then use the ERS API to download the results that were not sent to them. To facilitate the experimenters, the EMC also displays relevant information to the experimenters so that they can use the information to call the ERS APIs. This information mainly includes the JOBID and the EMOID. The analytics tool, however due to its asynchronous behaviour, stores the results in the ERS directly and experimenters are advised to frequently check back if the results are available in the ERS or not. ERS mainly has a POST and a GET API that enables the EEE or the analytics tool to send the result set to the ERS internal repository and experimenters to retrieve the result sets. The GET API of the ERS is only made public. Once the experimenters download the result sets, this GET API also deletes the result sets from the repository. It should also be noted that the EaaS Platform Portal is a one stop shop that integrates all the UIs relating to the experiment development, deployment and management services for easy access.

In order to successfully execute the EDSpec, experimenters should follow best practices that enable error free execution of the experiment. These best practices include: correctly specifying text tags, setting scheduling parameters such that it does not overload the system (i.e. not specifying execution to happen every second, writing queries that are not generic, or writing queries that result in huge datasets), providing the correct location where the data should be sent, and respecting the ontology structure in the SPARQL query.

The above-described workflow is one way of executing the experiments using the tools provided by the EaaS Platform. However, an experimenter can create their own EEE like tool using the APIs of the EEE, ERM, ERS and Analytics tool and execute that component on their side. Nonetheless, if an experimenter does not want to use the related APIs, they can create their own tools to call the IoT-Registry API and retrieve IoT data. As experimenters then can define their own workflow, describing them is out of the scope of this paper.

## VI. FIESTA-IoT EXPERIMENTATION VALIDATION AND EVALUATION

In this section we present the validation and evaluation of the instantiation of the platform design that has been created within the H2020 FIESTA-IoT project.[1] The instantiation of the IoT EaaS Platform in the so called FIESTA-IoT Platform, and the appealing results of its evaluation implicitly validates the adequacy of the design principles and the specification of the different building blocks that have been presented in the previous sections of the paper.

We carried out both qualitative and quantitative evaluation of the FIESTA-IoT platform in order to demonstrate that the hypotheses of this paper are correct. In particular, the evaluation focuses on the following three contributions:

1. We carry out a case-study based evaluation to show that the FIESTA-IoT platform supports semantically interoperable and testbed agnostic access to IoT data in order to allow cross-domain experimentation.
2. We performed a user-study, where external researchers and developers with access to the FIESTA-IoT platform performed experiments. Qualitative data from a questionnaire considers the extent to which the platform provides a usable and valuable tool in the development lifecycle.
3. Finally we carry out a performance evaluation of the platform. The quantitative data demonstrates that the platform is performant to users' needs and scales to increasing number of experimenters.

### A. EXPERIMENTAL SETUP

The following documents the instantiation of the FIESTA-IoT platform. The following core components and tools of FIESTA-IoT were deployed and secured on three virtual machines with the following characteristics:

- Core VM (32GB RAM, 12 vCPU cores and 1615GB disk space, Ubuntu v14.04): hosts the central IoT-Registry, the security components and all FIESTA-IoT tools and services (highlighted in Figure 2).
- Monitoring VM (16GB RAM, 8 vCPU cores and 160GB disk space, Ubuntu v14.04): hosts a Graylog server[7] for monitoring and analysing the FIESTA-IoT platform.
- Certification VM (8GB RAM, 4 vCPU cores and 80GB disk space, Ubuntu v14.04): hosts the FIESTA-IoT certification portal[8] with the tools used by new testbeds to test they are ready to join the platform.

With the platform deployed, the next step was to integrate cross-domain testbeds to provide the actual platform data to be used by experiments. In the first phase, four testbeds were integrated. This was followed by the integration of six further testbeds. The four initial testbeds are: i) *SmartSantander*, a large-scale Smart City deployment containing >3000 fixed and mobile sensors for environment, traffic, and crowd-sensing; ii) *SmartICS*, a Smart Building Environment, with >600 indoor sensors, iii) *SoundCity*, a large-scale

[7] https://www.graylog.org/
[8] http://certificate.fiesta-iot.eu/

crowd-sensing testbed with sensors on phones measuring noise, proximity, speed, location; and iv) *CABIN*, an indoor and outdoor smart building Smart Environment deployment with ∼200 sensors.

In order to enlarge the value of the offer and also to proof the adequacy of the solutions designed to enable interoperability among heterogeneous IoT platforms, two open calls for testbed integration were conducted. As a result of these Calls, seven more testbeds were selected.

The main aim of federating more IoT testbeds and not restricting it to the original four ones is to challenge the platform design. This way tuning of that design can be made by following the lessons learnt and best practices that can only be elicited from actual implementation. Moreover, addition of more application domains also brings further challenges that were not initially considered as they were not present in the initial set of testbeds. This selection was based on the following criteria: (1) Usefulness; (2) Complementarity; (3) Sustainability; (4) Technical competence; and (5) Feedback.

From a technical standpoint, each testbed implemented the testbed TPS component based upon the skeleton[9] to align with core ontology underpinning the semantic interoperability of the platform; in this case, we used the FIESTA-IoT ontology [17].

The complete description of the FIESTA-IoT ontology is out of the scope of this paper. A complete specification of the FIESTA-IoT ontology is defined in [17]. It is important to emphasize that this ontology is the baseline for the interoperability of the heterogeneous testbeds and IoT platforms that are federated in the FIESTA-IoT Platform. The different testbeds have to converge for participating in the federation and they use this ontology as the reference for this convergence. Precisely this is the main reason why the ontology has been kept simple as a design decision.

Yet another important design consideration has been the reuse, as much as possible, of already well-established concepts in the ontology. In this sense, for the core ARM concepts, the FIESTA IoT ontology has taken the IoT-lite ontology [36], a lighter version of the IoT-A ontology [37]. For the observations aspect, which is not correctly captured by IoT-Lite, the SSN ontology has been used. This ontology is specially chartered to describe sensors and observations, and related concepts. Finally, the phenomena and units of measurement related concepts have been incorporated to the FIESTA-IoT ontology through the M3-Lite taxonomy. This taxonomy has been created by integrating and aligning already existing ontologies in order to homogenize the existing scattered environment in which a quite large number of similar ontologies define the same concepts in an overlapping manner.

### B. CASE STUDY EVALUATION

To evaluate the FIESTA-IoT platform, we use a case-study based methodology. That is, we consider particular use cases where FIESTA-IoT has been applied and observe the extent to which these cases show the following hypotheses.

1. The FIESTA-IoT platform can be used to perform IoT experiments atop semantically interoperable data; thus facilitating the testing of solutions with a horizontal approach
2. The FIESTA-IoT platform supports both experimentation and technology maturation under realistic conditions in real world experimental deployments as part of the innovation lifecycle.

#### 1) ENMONITOR CASE STUDY

In this case study we carried out an experiment to develop a tool (named EnMonitor) to leverage cross-domain IoT data from multiple IoT testbeds as defined in the FIESTA-IoT platform previously described. The purpose of EnMonitor is to display in an intuitive manner near real-time information about the environment based upon data from all around the globe.

EnMonitor provides an easy-to-use web-based graphical interface where users can pinpoint to concrete regions on a map, select among different environmental phenomena and view different metrics (e.g. heatmap). To be specific, EnMonitor uses the data available to enable the users to do IoT resource discovery, perform observation harvesting, view different statistics and view aggregated environmental condition information.

The application is meant to be a proof-of-concept for the key added-value of the IoT EaaS Platform, which is to allow to access in a common way to multiple platforms that offer different IoT services targeted for different applications. Thus, when developing the EnMonitor application (as any developer would do with her application) the EaaS API has been used instead of having to learn and adapt the application to each of the APIs from the underlying IoT platforms. Examples of experiments and applications that have actually leveraged this ability from the FIESTA-IoT Platform can be found at http://fiesta-iot.eu/index.php/fiesta-experiments/.

EnMonitor and its interactions with the FIESTA-IoT platform is shown in Fig. 8. Thus the purpose of the experiment is twofold: i) to evaluate if the tool successfully interoperates with multiple heterogeneous data sources and is performant in terms of real-time data provision and responsiveness; and ii) real-world conditions support technology maturation to improve the tool towards offering citizens a holistic view of the environment around them and enabling policymakers to take advantage of federated data to complement their legacy decision tools.

EnMonitor was developed to use the APIs of the IoT-registry tool. Based on the user interactions from the GUI it queries the IoT-registry to obtain the results from the federated testbeds.

***Analysis***. EnMonitor successfully developed a performant tool using heterogeneous IoT data from real-world deployed sensors. The FIESTA-IoT platform provided a simple method to integrate these semantically interoperable data in

[9]https://github.com/fiesta-iot/testbed.tpi

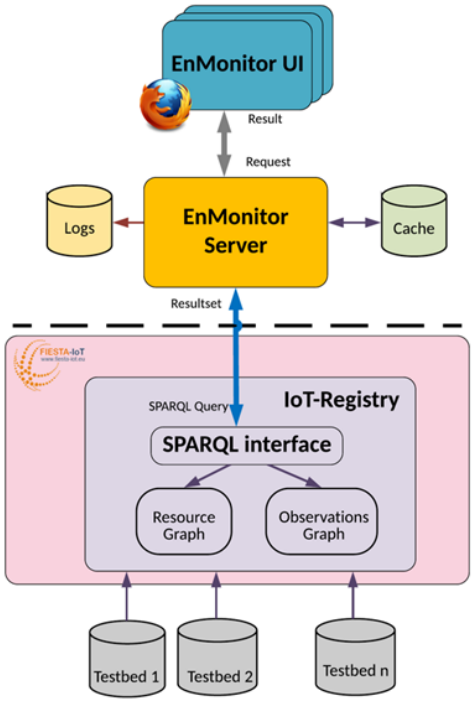


**FIGURE 8.** EnMonitor interactions with FIESTA-IoT platform (Source [38]).

a transparent way; as such the tool is also easily extensible to consider new environmental phenomena as and when new testbeds and sensor data are integrated into the FIESTA-IoT platform. Hence, this shows the benefits the EaaS approach atop semantically interoperable data provides. The access to real-world sensor data also supported the quick maturation of the tool (as opposed to working with simulated data)—that is, the tool could be validated in the real environment with real data.

#### 2) EXTERNAL EXPERIMENTERS CASE STUDY

In this case study we made the FIESTA-IoT platform available to use by external experimenters. These were recruited using an open call funding competition for 24 experiments from academic researchers and/or commercial organisations. External parties submitted experiment proposals that were independently evaluated, and the winners obtained money to carry out their proposed experiment over 6 months. That is the funding to implement any technology and perform experiments or technology validation. All 23 experiments were successfully developed and deployed using the FIESTA-IoT platform; the following summarizes the key outcomes:

- The experiments covered multiple IoT domains: 6 smart city experiments, 4 smart energy, 2 smart agriculture, 6 data science, 1 data representation, 3 IoT platform, and 2 IoT Networking experiments.
- The experiments leveraged data from multiple testbeds. 10 experiments used 2 testbeds, 4 experiments used 3 testbeds, and there were 3 experiments that used 4, 5 and 6 testbeds respectively.
- The experiments covered different stages in the innovation lifecycle: 11 carried out scientific research, and 13 technology innovation and validation experiments (prior to market).

***Analysis***. The results of these external experiments demonstrate that the results achieved in the EnMonitor experiment case study have been replicated by third party users of FIESTA-IoT; that is, these experiments have also benefited from testbed-agnostic access to semantically interoperable data from real-world IoT sensor deployments. Note, further user-based evaluation of the platform (in terms of their experience with ease-of-use and value obtained) is described in Section VI.C.

### C. USER EVALUATION of EaaS

In this section we provide a more detailed evaluation of the usefulness of the FIESTA-IoT platform.

#### 1) METHODOLOGY

As independent users of the FIESTA-IoT platform, the 23 selected external experimenters were asked, at the end of their 6 months experience, to fill a questionnaire and a KPIs evaluation form. The purpose of the questions were to evaluate the users' opinions about both the functions provided by the platform and also the quality and performance they observed. The 23 experimenters did not use the FIESTA-IoT platform over the same time (or conditions). There were two usage waves 6 months apart: the 1st wave consisted of 6 experimenters, considered as alpha testers, whilst 17 experimenters, seen as beta testers, participated to the 2nd wave.

#### 2) RESULTS

The questionnaires used a Likert scale to obtain the experimenters attitude to questions about their usage of the FIESTA-IoT platform; feedback was returned in form of score between 1 and 5—in this scale, 1 stands for ''very poor'' and 5 ''excellent''. As shown in Fig. 9 the experience generally improved from the 1st to the 2nd wave. The stability and usability of the platform reached a high level of satisfaction (around 4) whilst the general performances, reaching a steady 3.5, might be considered for improvement. The performance and availability of the portal, used for designing experiment, have significantly improved between the two waves to reach grades of circa 4.5.

The process of integrating and deploying their experiments using the tools available have been considered satisfactory with a grade of almost 4. The questionnaire considered the effort required by the external experiments. Here, the results reported that an initial exploration of the platform needs less than 15 days whilst for a full implementation and integration of the experiment between 30 and 60 days development is needed.

Also the tools offered by FIESTA-IoT have been assessed quite satisfactory going over the 3.5. As explained in the previous sections, users had the possibility to implement the

experiment either as direct calls to the APIs exposing the data and the resources, or via a textual definition of the experiment, both of the two approaches expect the creation of SPARQL queries. As reported in Fig. 9 the process of creating the SPARQL query is considered "*very good*", enhancing substantially between the 1st and 2nd wave.

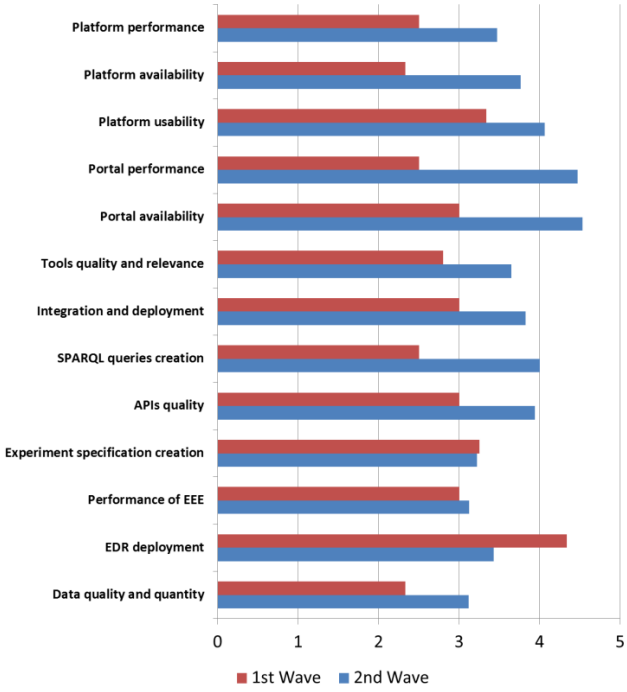


**FIGURE 9.** Experience and satisfaction with FIESTA-IoT.

Whilst the assessment of the APIs is improved, the process and the tools for the creation of the experiment within the portal have not been always satisfactory. We believe that this is not due to the particular deficiency of the tools but rather to a higher familiarity with the APIs approach. As a matter of fact the number of users preferring the APIs approach is 16, the users that preferred the experiment description approach is 5 and only 2 users have expressed no preference among them.

The experimenters were also asked also about the market appeal of the offered platform and the results are shown in Fig. 10. As it can be seen, 55.5 % of the users would pay for the service with different formulas: pay-per-use, return of activity, one off charge, subscription basis. A user, instead, would consider paying if the return of investment is attractive for their business. Finally among the 33.3% not willing to pay, most of the users consider that such asset should be maintained by public institutions, whereas only one user would consider payment as option only after improvements of the platform. Such results highlight that the experimenters identify the importance of the capabilities provided by the platform.

The weakest point of the platform seems to be the quality and the quantity of the data. This point is not directly affected

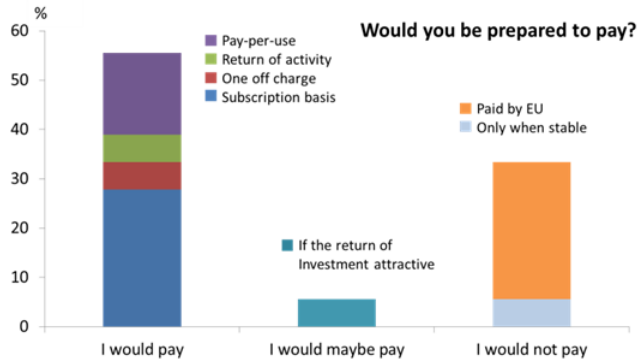


**FIGURE 10.** Market appeal of FIESTA-IoT.

by the platform concept and functionalities but rather by the quality of testbed deployment integrated.

Finally, considering the questions concerning overall satisfaction with the FIESTA-IoT platform, 15 of the experimenters responded with a "full satisfaction", 8 with "a partial satisfaction" and none of them responded with complete negative feedback. We believe that enhancing the integrated testbeds, by the number or by the quality, would address the main roots of dissatisfactions. In any case, all of the users stated they would recommend the FIESTA-IoT platform to other experimenters.

### D. PERFORMANCE EVALUATION

In the previous section we have analyzed the functional evaluation of the platform, mainly based on the experience reported and gathered from experimenters. However, in order to complete the evaluation of the FIESTA-IoT Platform considering the technical aspects, we have also performed a performance assessment through the analysis of the time that the Platform, more specifically the IoT-Registry component, took to reply to the queries that it received while the experiments were conducted. Moreover, the analysis also presents the demand that the FIESTA-IoT Platform was handling in terms of requests per unit of time.

The analysis focuses on the SPARQL query response time because of two main reasons. On the one hand, this is the most time consuming procedure. When the experimenter is using the functionalities of the Resource Broker (i.e. the other alternative to retrieve data from the underlying resources), the IoT-Registry basically acts as a proxy, thus, introducing only some milliseconds of processing delay. On the other hand, most of the experimenters used the SPARQL endpoint of the IoT-Registry to retrieve data.

The analysis was carried out between 6th February 2018 and 15th March 2018. Fig. 11 shows the amount of queries that the IoT-Registry received each day. It is important to highlight that the IoT-Registry is not only serving the experimenters' demands but, at the same, it has to keep storing the observations that are constantly coming from the underlying testbeds. In average the FIESTA-IoT Platform received 13,000 queries per day, which is equivalent to 9 queries per minute.

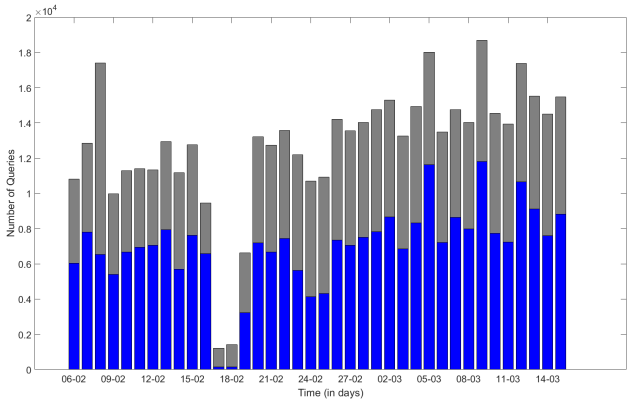


**FIGURE 11.** Number of queries received by IoT-Registry.

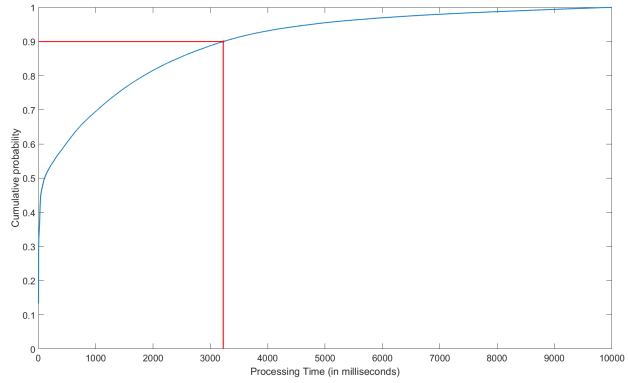


**FIGURE 12.** Cumulative distribution function of SPARQL queries processing times.

Fig. 11 shows that the overall workload that the Platform had to handle was quite steady, except for 17th and 18th February, when there is an abrupt reduction on the number of queries. Maintenance and updates tasks were scheduled those days and only the internal testing queries were received.

As it can be seen in Fig. 12, 90% of the queries are handled in less than 3.23 seconds while the amount of queries taking more than 10 seconds is negligible. Moreover, it is interesting to highlight that more than half of the queries were responded in less than 100 milliseconds. In this respect, it has to be noted that when the SPARQL query was wrongly formatted (e.g. because of experimenter mistake) the processing delay was 0 as the Data Endpoint of the IoT-Registry immediately detected the syntactic errors.

Whereas these wrongly formatted queries accounted for around 10% of the total, as it can be seen in Fig. 13 still most usual response times are below 40 milliseconds. It is important to note that the Probability Density shown in Fig. 13 excludes queries solved in 0-time.

Taking into account these results, we can conclude that the FIESTA-IoT Platform, which is a running instance of the IoT EaaS Platform described in this paper, is showing a quite good performance which should fulfil the needs from any experiment or application requesting semantically interoperable data from it.

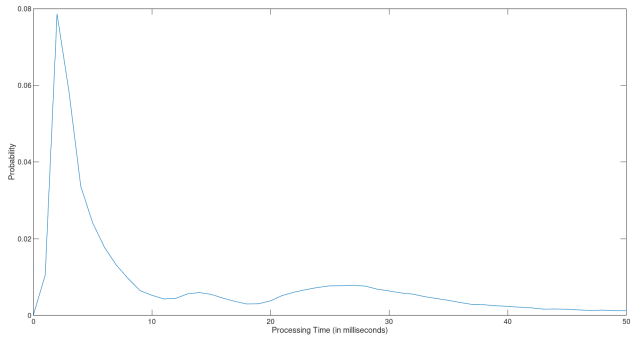


**FIGURE 13.** Probability density function of SPARQL queries processing times.

## VII. CONCLUSIONS

Enabling seamless experimentation over real-world testbeds represents a major advantage to underpin research and innovation aimed at having direct and fast impact in our everyday lives. This paper has presented the design considerations of an IoT EaaS Platform and the specification of its building blocks. This platform has been instantiated in a cloud-based environment and it is currently integrating 11 different IoT testbeds with over 2,500 sensors in total. These testbeds have different application domains, from smart cities to maritime environmental monitoring, but a common denominator, all of them are deployed in real-world environments.

The IoT EaaS Platform exposes a unique set of tools and APIs aimed at reducing the experimenters' effort to build and run experiments that might expand across federated IoT deployments. Throughout the paper, the experimentation workflow is described together with the platform's components that enables it. In this sense, the EaaS paradigm enabled by the IoT EaaS Platform described in the paper ranges from plain access to raw data and/or services offered by any of the underlying testbeds to autonomous scheduling and execution of experiments involving added-value analytics and/or reasoning techniques.

Taking advantage of the actual instantiation of the Platform design and the integration of real IoT testbeds, the design has been refined together with the interfaces and models supporting IoT testbeds semantic alignment and interoperability. The resulting platform increases scale, heterogeneity, experimentation realism and cross-domain innovation as more testbeds are joining the federation.

In order to proof the validity and appropriateness of the proposed design, the instance of the IoT EaaS Platform that have been developed in the framework of the EU H2020 FIESTA-IoT project has been subject of both qualitative and quantitative evaluation. This evaluation has been done in the framework of actual experimental-based research and innovation made over the instantiated IoT EaaS Platform. The results have shown that the proposed design have fulfilled experimentation requirements demonstrating excellent performance even under heavy duty.

Future work includes the continuous extension of this instance of the Platform through the addition of more testbeds

as well as the support for publish-subscribe interactions with the platform so that experimenters can be notified upon occurrence of an event in which they are interested. Currently, the experimenter can only retrieve data upon direct request.

## ACKNOWLEDGMENT

The authors would also like to thank the FIESTA-IoT consortium for fruitful discussions.

## REFERENCES

[1] S. H. Thomke, *Experimentation Matters: Unlocking the Potential of New Technologies for Innovation*. Boston, MA, USA: Harvard Business School Press, 2003.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[3] A. Rachedi, M. H. Rehmani, S. Cherkaoui, and J. J. P. C. Rodrigues, "IEEE access special section editorial: The plethora of research in Internet of Things (IoT)," *IEEE Access*, vol. 4, pp. 9575–9579, 2017.

[4] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. McCann, and K. Leung, "A survey on the IETF protocol suite for the Internet of Things: Standards, challenges, and opportunities," *IEEE Wireless Commun.*, vol. 20, no. 6, pp. 91–98, Dec. 2013.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[6] P. Neirotti, A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano, "Current trends in smart city initiatives: Some stylised facts," *Cities*, vol. 38, pp. 25–36, Jun. 2014.

[7] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[8] J. M. Hernández-Muñoz *et al.*, *Smart Cities at the Forefront of the Future Internet*. Berlin, Germany: Springer, 2011, pp. 447–462.

[9] L. Zhang *et al.*, "A remote medical monitoring system for heart failure prognosis," *Mobile Inf. Syst.*, vol. 2015, Sep. 2015, Art. no. 406327.

[10] S. M. Riazul Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, Jun. 2015.

[11] K. Zheng, S. Zhao, Z. Yang, X. Xiong, and W. Xiang, "Design and implementation of LPWA-based air quality monitoring system," *IEEE Access*, vol. 4, pp. 3238–3245, 2016.

[12] K.-L. Tsai, F.-Y. Leu, and I. You, "Residence energy control system based on wireless smart socket and IoT," *IEEE Access*, vol. 4, pp. 2885–2894, 2016.

[13] J. Wan, M. Yi, D. Li, C. Zhang, S. Wang, and K. Zhou, "Mobile services for customization manufacturing systems: An example of industry 4.0," *IEEE Access*, vol. 4, pp. 8977–8986, 2016.

[14] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, "A survey on facilities for experimental Internet of Things research," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 58–67, Nov. 2011.

[15] L. Sanchez *et al.*, "SmartSantander: IoT experimentation over a smart city testbed," *Comput. Netw.*, vol. 61, pp. 217–238, Mar. 2014.

[16] A.-S. Tonneau, N. Mitton, and J. Vandaele, "A survey on (mobile) wireless sensor network experimentation testbeds," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, May 2014, pp. 263–268.

[17] R. Agarwal *et al.*, "Unified IoT ontology to enable interoperability and federation of testbeds," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 70–75.

[18] G. Coulson *et al.*, "Flexible experimentation in wireless sensor networks," *Commun. ACM*, vol. 55, no. 1, pp. 82–90, Jan. 2012.

[19] C. Adjih *et al.*, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 459–464.

[20] W. Vandenberghe *et al.*, "Architecture for the heterogeneous federation of future internet experimentation facilities," in *Proc. Future Netw. Mobile Summit*, 2013, pp. 1–11.

[21] M. Berman *et al.*, "GENI: A federated testbed for innovative network experiments," *Comput. Netw.*, vol. 61, pp. 5–23, Mar. 2014.

[22] A. Misra and R. K. Balan, "LiveLabs," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 17, no. 4, pp. 47–59, Dec. 2013.

[23] G. Cardone, A. Cirri, A. Corradi, and L. Foschini, "The participact mobile crowd sensing living lab: The testbed for smart cities," *IEEE Commun. Mag.*, vol. 52, no. 10, pp. 78–85, Oct. 2014.

[24] A. Willner, M. Giatili, P. Grosso, C. Papagianni, M. Morsey, and I. Baldin, "Using semantic Web technologies to query and manage information within federated cyber-infrastructures," *Data*, vol. 2, no. 3, p. 21, Jun. 2017.

[25] M. Avgeris, N. Kalatzis, D. Dechouniotis, I. Roussaki, and S. Papavassiliou, *Semantic Resource Management of Federated IoT Testbeds*. Cham, Switzerland: Springer, 2017, pp. 25–38.

[26] I. Tachmazidis *et al.*, *A Hypercat-Enabled Semantic Internet of Things Data Hub*. Cham, Switzerland: Springer, 2017, pp. 125–137.

[27] A. D'Elia, F. Viola, L. Roffia, P. Azzoni, and T. S. Cinotti, "Enabling interoperability in the Internet of Things: A OSGi semantic information broker implementation," *Int. J. Semantic Web Inf. Syst.*, vol. 13, no. 1, pp. 147–167, 2017.

[28] R. Petrolo, V. Loscri, and N. Mitton, "Towards a smart city based on cloud of things—A survey on the smart city vision and paradigms," *Trans. Emerg. Telecommun. Technol.*, vol. 28, no. 1, p. e2931, Jan. 2017.

[29] A. Palavalli, D. Karri, and S. Pasupuleti, "Semantic Internet of Things," in *Proc. IEEE 10th Int. Conf. Semantic Comput. (ICSC)*, Feb. 2016, pp. 91–95.

[30] A. Bassi, *Enabling Things to Talk*. Cham, Switzerland: Springer, 2013.

[31] S. Cox, "Observations and measurements," *Open Geospatial Consortium Best Pract. Document*, no. 05-087r4, p. 21, 2006.

[32] *FIWARE Data Models*. Accessed: May 24, 2018. [Online]. Available: https://www.fiware.org/developers/data-models/

[33] K. G. Clark, K. Grant, and E. Torres, *SPARQL Protocol for RDF*, document 20080115, W3C Recommendation, 2008. [Online]. Available: http://www.w3.org/TR/rdf-sparql-protocol/

[34] J. Lanza, L. Sanchez, D. Gomez, T. Elsaleh, R. Steinke, and F. Cirillo, "A proof-of-concept for semantically interoperable federation of IoT experimentation facilities," *Sensors*, vol. 16, no. 7, p. 1006, Jun. 2016.

[35] A. Ahrabian, S. Kolozali, S. Enshaeifar, C. Cheong-Took, and P. Barnaghi, "Data analysis as a Web service: A case study using IoT sensor data," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 6000–6004.

[36] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "IoT-lite: A lightweight semantic model for the Internet of Things," in *Proc. Int. IEEE Conf. Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People, Smart World Congr.*, Jul. 2016, pp. 90–97.

[37] S. De, P. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for the Internet of Things," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, 2011, pp. 949–955.

[38] R. Agarwal, D. Gomez, J. Lanza, L. Sanchez, N. Georgantas, and V. Issarny. *EnMonitor: Experimentation Over Large-Scale Semantically Annotated Federated IoT Data Environment*. Accessed: May 24, 2018. [Online]. Available: https://hal.inria.fr/hal-01579413v2

**JORGE LANZA** received the Ph.D. degree in telecommunications engineering from University of Cantabria in 2014. He has participated in several research projects, national and international, with both private and public funding. He is a Senior Researcher at the Network Planning and Mobile Communications Laboratory, University of Cantabria, Spain. His current research is focused on IoT infrastructures toward federating deployments in different locations using semantics technologies. In addition, his work has included combined mobility and security for the wireless Internet.

**LUIS SÁNCHEZ** received the Telecommunications Engineering and Ph.D. degrees from the University of Cantabria, Spain, in 2002 and 2009, respectively. He is currently an Associate Professor at the Department of Communications Engineering, University of Cantabria. He is active on IoT-enabled smart cities, meshed networking on heterogeneous wireless scenarios, and optimization of network performance through cognitive networking techniques. He has a long research record involved in projects belonging to the fifth, sixth, seventh, and H2020 EU Framework Programs. He has authored over 60 papers at international journals and conferences and co-authored several books. He often participates in panels and round tables discussing about innovation supported by IoT in smart cities. He also acts as an expert for French ANR (Agencie National Recherche) and Italian MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca) reviewing and evaluating research and development proposals.

**JUAN RAMÓN SANTANA** received the Telecommunication Engineering degree from the University of Cantabria (UC) in 2010. He is currently a Research Fellow with the Network Planning and Mobile Communications Laboratory, Telecommunication Research Group, UC. Prior to his current occupation, he did an internship at the University of Strathclyde, Glasgow, where he was involved in IoT solutions. He has also been involved in several projects, such as SmartSantander, EAR-IT or FESTIVAL, and European collaborative projects related to the Smart City paradigm and the Internet of Things. Among his research interests are wireless sensor networks, M2M communications, and mobile phone application research.

**RACHIT AGARWAL** received the Ph.D. degree in computer science and telecommunications from the University of Pierre and Marie Curie, Paris, in 2013, with the laboratory situated at Telecom SudParis. He holds a post-doctoral/Researcher Engineer position at Inria-Paris and is associated to the MiMove Research Team within Inria. His research interests mainly span the areas related to ICT, especially relating to Internet of Things (IoT), human mobility aspects, semantic technologies, and network science. In the past, he has been associated to several projects and has been the Co-PI of Inria's Sarathi Associate Team. He has won the 2015 Semantic Web Challenge. He is currently a reviewer to many international journals and conferences, and has served as a PC Co-Chair for the Advanced and Trusted Internet of Things and Smart City Track at the 12th IEEE International Conference on Advanced and Trusted Computing (ATC 2015), Beijing, China, in 2015.

**NIKOLAOS KEFALAKIS** received the Diploma degree in electronic computing systems from the Higher Technological Educational Institute of Piraeus and the M.Sc. degree in information technology and telecommunications from the Athens Information Technology. Since 2008, he has been involved in the IoT Systems Group of AIT in the area of Intelligent RFID Systems and Internet of Things (IoT), where he is a Senior Researcher. He has been involved in various EU projects and more specifically in the context of ASPIRE and OpenIoT FP7 projects. He is the Manager, System Architect, and Developer Lead of the AspireRFID OS Project (http://wiki.aspire.objectweb.org/) and the multiple award winning OpenIoT OS project (https://github.com/OpenIotOrg/openiot). His main area of technical expertise is IoT systems, auto-ID technologies (RFID, barcodes…), semantic sensor networks, multitier architecture systems, enterprise systems, and embedded and electronics digital systems.

**PAUL GRACE** received the B.Sc. degree in computer science from the University of York, U.K., and the M.Sc. and Ph.D. degrees in distributed systems from Lancaster University, U.K. He is currently a Senior Researcher with the School of Electronics and Computer Science, University of Southampton. His research interests are in: secure distributed systems, privacy engineering, middleware, and software modelling.

**TAREK ELSALEH** received the B.Eng. degree (Hons.) in electronic engineering from Oxford Brookes University and the M.Sc. degree in communications, networks, and software from the University of Surrey. He is a Research Fellow and a Systems Developer at the Institute for Communication Systems, University of Surrey. His previous work experience and degree projects have revolved around sensors and data management. His background includes sensor system design and instrumentation, localization, embedded software development, Web development, media content adaption, Internet of Things, Web of Things, and Linked Open Data. He has been involved in IoT EU FP7/H2020 and U.K. research projects, including SENSEI, IoT-A, FIWARE, and currently in FIESTA-IoT, NHS Testbeds, and ACTIVAGE.

**MENGXUAN ZHAO** received the Ph.D. degree in computer science from the University of Grenoble. In her thesis, she brought the classical discrete control theory into the new application domain of the IoT using semantic techniques. She joint Easy Global Market in 2015 after three years of work of thesis preparation in Orange Labs, Grenoble. She mainly involved in European research projects, including Fiesta-IoT (semantic interoperability of testbeds), Festival (EU-JP, interoperability and federation of ICT services and testbeds), and several 5G-related projects (5GinFIRE, 5GTANGO). Her research interests include IoT, data interoperability, and testing methodology. She also participates and contributes to standardization works, including participation and organization of plugtest events.

**ELIAS TRAGOS** received the M.B.A. degree in business administration in techno-economics and the Ph.D. degree in wireless communications. He has been actively involved in many EU and national research projects as a Researcher, Technical Manager, and Project Coordinator. He is a Research Project Manager at the Insight Centre for Data Analytics, University College Dublin, Ireland. He has authored or co-authored over 70 peer-reviewed conference and journal papers, receiving over 1800 citations. His research interests lie in the areas of wireless and mobile communications, Internet of Things, cognitive radios, network architectures, fog computing, security, privacy, and recommender systems.


**HUNG NGUYEN** received the B.S. degree in information technology from the Hue University of Science, Vietnam, in 2007. He is an Experienced Software Developer, having involved in multiple small to large software developing companies as a Java, .Net, and Cloud developer. He was a Team Leader, Software Architect, Scrum Master, and Software Designer. He is currently a Researcher at the Insight Centre for Data Analytics, National University of Ireland, Galway, where he is involved in several EU funded projects. His research interests include Internet of Things, artificial intelligence, micro-services, deep learning, and security and privacy.

**FLAVIO CIRILLO** received the master's degree in computer engineering from the University of Naples Federico II in 2014. He is a Research Scientist with the IoT Research Team, NEC Laboratories Europe, Germany. His research focus is in the Internet-of-Things analytics and platforms field, especially scalability and federation aspects and semantics enablement, in the scenario of Smart Cities. He is currently one of the main developers and maintainers of the IoT Backend layer of the IoT Architecture of FIWARE. He was involved in IoT-related European research project, such as FIWARE, FIESTA-IoT, AUTOPILOT, SynchroniCity, and others. He is currently part of the Information Technology and Electrical Engineering Ph.D. Programme XXXIII Cycle at the University of Naples Federico II.

**RONALD STEINKE** is currently pursuing the computer engineering degree with Technische Universität Berlin with a focus on network technologies. From 2009 to 2012, he was a Student Researcher at the TKN Institute, Technische Universität Berlin, where he was involved in the area of network coding and future Internet. In 2012, he joined the NGNI Department, Fraunhofer FOKUS Institute, as a Student Researcher. At NGNI, he was helping developing the OpenMTC Platform and was involved in the area of M2M and IoT. In 2014, he wrote his Diploma Thesis at the NGNI Department with the title Design and Implementation of an ETSI M2M Compliant Control Framework for Smart Grids and graduated as a Graduate Engineer. In 2014, he joined the Chair Next Generation Networks at Technische Universität Berlin. In 2015, he joined Fraunhofer FOKUS continuing developing the OpenMTC Platform and involved in several projects.

**JOHN SOLDATOS** received the Ph.D. degree in electrical computer engineering from the National Technical University of Athens in 2000. He was an Adjunct Professor at Carnegie Mellon University from 2007 to 2010. Since 2006, he has been an Associate Professor at Athens Information Technology. Since 2014, he has been an Honorary Research Fellow at the University of Glasgow, U.K. From 2014 to 2016, he was a member of the European Crowdfunding Stakeholders Forum, and from 2012 to 2015, he was the Coordinator of the IoT Identification, Naming and Discovery Group, European Internet-of-Things Research Cluster. He has had a very active role in many EC co-funded research and development projects, in the scope of the FP6, FP7, and H2020 programmes, including several projects in pervasive computing, cloud computing, Internet-of-Things, and BigData. He has also participated in major enterprise consulting projects as a principal business consultant in the areas of ICT, industry, energy, and healthcare. He is co-founder of the open source platforms OpenIoT (https://github.com/OpenIotOrg/openiot) and AspireRFID (http://wiki.aspire.ow2.org). He has authored or co-authored over 180 articles in international journals, books, and conference proceedings.