Invited Paper

FESTIVAL: Design and Implementation of Federated Interoperable Smart ICT Services Development and Testing Platform

Toyokazu Akiyama[1,a)] Shuuichirou Murata[2] Kiichi Tsuchiya[3] Teruaki Yokoyama[1] Martino Maggio[4] Giuseppe Ciulla[4] Juan Ramón Santana[5] Mengxuan Zhao[6] Jander Botelho Do Nascimento[7] Levent Gürgen[7]

Received: October 4, 2016, Accepted: December 26, 2016

Abstract: FESTIVAL EU-Japan collaborative project aims at federating existing Smart ICT testbeds of different nature to provide a platform for developing and testing emergent Smart ICT services. The federation of testbeds covering heterogeneous domains has been a great challenge and FESTIVAL provides a uniform access to different resources, such as Open Data resources, IoT devices, IT resources and Living Labs. In this paper, design and implementation of the current FESTIVAL platform are introduced with approaches to federate and interoperate existing resources. Integration of all the components, including the existing testbeds, will also be described to finalize and validate the federation.

Keywords: testbed federation, Smart ICT experiment platform

1. Introduction

The development of the Internet of Things is set to have a strong impact on many aspects of society. Testbeds and experimental facilities, both of small scale and up to city scale, will be an essential enabler to facilitate and validate the development of this vision. There have been long years of research work in Europe and Japan on federation of testbeds and more recently on IoT testbeds. FESTIVAL aims at leveraging those testbeds by a federation approach where experimenters can seamlessly perform their experiments taking benefit of various software and hardware enablers provided both in Europe and in Japan. Facilitating the access to those testbeds to a large community of experimenters is a key asset to the development of a large and active community of application developers, necessary to address the many challenges faced by European and Japanese societies.

The main objectives of the project can be summarized by the following list:

- Exploit existing European and Japanese assets to enable IoT data collection and processing.
- Enable federation and interoperability of the testbeds through a common API.
- Build services and experimentation over the federated testbeds cutting across various application domains (energy, building, shopping...).
- Offer access to external experimenters in both Europe and Japan through Experimentation as a Service model.
- Evaluate the technical, economic and societal performance of the experimentations.

The FESTIVAL federation will connect cyber world to the physical world, from large scale deployments at a city scale, to small platforms in lab environments and dedicated physical spaces simulating real-life settings. Those platforms will be connected and federated via homogeneous access APIs with an "Experimentation as a Service" (EaaS) model for experimenters to test their added value services.

In this paper, we will describe work performed during first two-year period of the project. Requirement analysis of experimentation platform will be described in Section 2 including the investigation of experiment use cases and existing testbed. And then, the FESTIVAL architecture designed based on the investigation will be described in Section 3. In Section 4, an evaluation framework for the platform which validates the level of requirement satisfaction will be introduced. Consideration of remaining issues will be discussed in Section 5. Finally, we will conclude this paper in Section 6.

2. Requirement Analysis of Experimentation Platform

As described in Section 1, the goal of FESTIVAL is to establish an experimentation platform. Since the platform is required to

[1] Kyoto Sangyo University, Kyoto 603–8555, Japan
[2] Acutus Software, Inc., Meguro, Tokyo 153–0064, Japan
[3] JR WEST JAPAN COMMUNICATIONS COMPANY, Osaka 530–0003, Japan
[4] Engineering Ingegneria Informatica, Viale Regione Siciliana 7275, 90146 Palermo, Italy
[5] University of Cantabria, Plaza de la Ciencia s/n, Santander 39005, Spain
[6] Easy Global Market, Sophia-Antipolis, 06560 France
[7] Université Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France
[a)] akiyama@cc.kyoto-su.ac.jp

reduce the cost of experimenters by providing software and hardware enablers, first two objectives, "exploit existing assets" and "enable federation and interoperability of the testbeds through a common API," have priority to be achieved. However, an experimentation has extremely different requirements depends on its application domain and deployment scale. The investigation of use cases and existing assets are required to extract common API.

In the following sections, firstly, the existing testbeds federated to provide the common API will be introduced in Section 2.1. And then, the applications investigated in the FESTIVAL project for extracting common requirements will also be described in Section 2.2.

2.1 Existing Testbeds

There are ten testbeds that have been federated under the FESTIVAL EaaS platform. These testbeds can be classified depending on the four types of resources provided by the federated testbeds: Open Data resources, IoT devices, IT resources and Living Lab resources. The following sections describe each resource.

2.1.1 Open Data Platforms

The Open Data platforms are oriented to provide information following open data policies, where anyone can access a large amount of data for free to develop new services and applications. These Open Data platforms are provided by two European cities: Lyon and Santander.

The Metropole of Lyon's Open Data [1] is a platform that aggregates all the existing sources of data in Lyon to make them public for the citizens and developers. It includes several types of data, but most of them are related to the mobility sector: availability of shared bicycles, highway events or real-time traffic.

The Santander Open Data is a platform deployed by the Santander City Council that offers official and public data in many exploitable formats. Among the datasets available, more than 88 catalogues at the time being, we can find traffic related datasets, including parking information and public transportation real-time data, environmental data or culture events.

2.1.2 IoT Devices

The IoT devices in FESTIVAL can be divided in two types: deployed sensors measuring multiple parameters, e.g., temperature, humidity or traffic; and actuators, programmable devices that are able to modify the environment remotely. In this sense, four different testbeds have been federated within the EaaS platform:

The ATR Data Centre is a datacentre facility funded by the Ministry of Environment in Japan. The goal of this testbed is to provide sensor data to experiment with and reduce the overall energy required by data centres.

SmartSantander [2] is an urban testbed deployed in the city of Santander. It is composed by more than 12,000 sensors deployed throughout the city, addressing multiple domains such as both, fixed and mobile environmental monitoring, traffic and outdoor parking monitoring and parks and gardens irrigation measuring.

The iHouse facility [3], an experimental smart house that includes many sensors and actuators, such as temperature/humidity sensors, wind speed/direction sensors, door close/open sensors, rain sensors, illuminometers, pressure sensors for chairs/sofas, window openers/closers, LED lights, and air conditioners. It has a possibility to execute HEMS experiments as described in Ref. [4].

Finally, the PTL testbed (recently renamed as IRT PULSE [5]), located in the CEA headquarters in Grenoble, which is equipped by various types of sensors and actuators from different providers communicating with heterogeneous protocols. Among these sensors and actuators can be found luminosity sensors, temperature sensors, smart plugs, contact sensors at doors, rolling shutters, blinds or dimming lamps. The testbed is dedicated to energy efficiency and intelligent home control experimentation.

2.1.3 IT Resources

The IT resources in FESTIVAL are considered all the resources that provides infrastructure to experimenters. More precisely, the testbeds that includes computational power, through available virtual machines; and SDN-based service orchestration, to carry out experiments modifying the network configuration. There are two testbeds within this group in FESTIVAL:

The JOSE testbed [6], a testbed providing a huge set of connected computer servers distributed throughout Japan. Furthermore, JOSE testbed provides advanced SDN capabilities and high speed network for interconnecting all the computational resources.

The Engineering FIWARE lab is an instance of the FIWARE stack in a cloud infrastructure based on OpenStack. This testbed uses a set of preconfigured virtual machine images to rapidly deploy instances of Generic Enablers from FIWARE. Therefore, users can easily deploy, configure and use them for experimenting.

2.1.4 Living Lab Resources

Last but not least, the Living Lab resource group is composed by those testbeds that provide end-user interactions as a resource for experimentation. These interactions are composed by surveys, feedback or co-creation meetings, where end-users can directly participate into the experiment with their own points of view and ideas.

Belonging to this group, in the EaaS platform we can find two testbeds: on the one hand, the TUBA Living Lab [7], that includes two areas for experimenting with new services and help developing new projects, from start-ups and small medium enterprises (SMEs) to large companies; on the other hand, The Lab., an innovative space placed in the Grand Front Osaka building. The Lab. [8] is an access-free space where all kind of public can experience the latest available technology, including state-of-the-art prototypes and the participation in innovative experiments.

2.2 Experiments Using the FESTIVAL EaaS Platform

Along with the development of the FESTIVAL EaaS platform, several experiments have been developed. These experiments are intended to serve as a showcase for future experimenters who will make future use of the platform. As aforementioned, the target experiments are divided in three main domains: Smart Energy, Smart Building and Smart Shopping. Although the experiments are mainly based on one of the domains, some of them covers another one. This situation is shown in **Fig. 1**.

Finally, the platform also includes not only resources related to

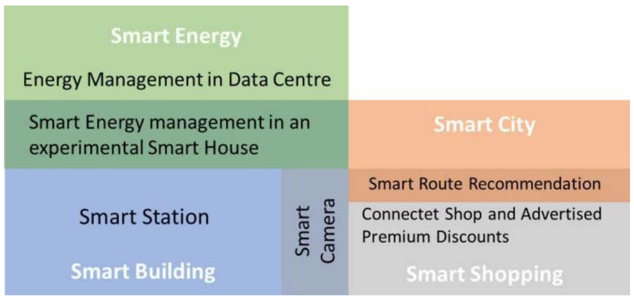

Smart Energy
Energy Management in Data Centre
Smart Energy management in an experimental Smart House
Smart City
Smart Route Recommendation
Connectet Shop and Advertised Premium Discounts
Smart Station
Smart Camera
Smart Building
Smart Shopping

Fig. 1 Experiments in FESTIVAL.

the previous domains, but also to the Smart City domain, which can be addressed by external experimenters to develop and deployed applications based on real-time data from urban sensors. The following sections describe some of the most representative use cases within each of the defined domains in FESTIVAL.

2.2.1 Smart Energy

The Smart Energy domain embraces all the experiments and applications that pursue an efficient energy management to reduce the energy consumption through the use of Internet of Thing sensors.

One of the experiments under this domain is the Data Centre Energy Management [9], [10]. This experiment aims at developing an effective prediction of temperature distribution in data centres, using machine learning techniques over sensor data. The final goal of this experiment is to reduce power consumption of air conditioners while keeping the system working under optimal conditions.

On the other hand, the Smart Energy Management in an experimental Smart House is being developed under the PTL premises, in the CEA headquarters in Grenoble. The experiment includes the use of multiple sensors and actuators from different vendors to be managed homogeneously under a controlled environment. As a result of this experiment, the deployment of a simple code allows the control of all the devices.

2.2.2 Smart Building

The Smart Building domain addresses all the experiment and applications using sensors and actuators deployed in buildings to research on automation techniques.

In this regard, the Smart Station experiment consists on the deployment of multiple sensors in two different train stations (Maya Station in Hyogo prefecture and Kameoka Station in Kyoto prefecture) in Japan, measuring the parameters such as Pollen, Particle Matter 2.5, vibration, acceleration, noise or temperature and humidity [11]. These sensors have been integrated using FIWARE [12] components and federated under the FESTIVAL EaaS platform. Some of the applications addressed is the possibility of advertising to the train passengers in real time about the status of the station using existing advertising spaces.

Furthermore, the Smart Camera experiments consist on the development of basic computer vision system. It uses a compact smart camera to gather specific features, which are later analyzed to evaluate, for instance, the number of people in a room. The main advantage of this Smart Camera experiment is the privacy-oriented development. In this sense, all the data provided by the Smart Cameras are already anonymized and available to be used by any experimenter without privacy concerns.

2.2.3 Smart Shopping

The Smart Shopping domain includes all the applications and experiments to encourage the relationship between customers and shop managers, through the use of novel technologies in the shopping areas, such as sensors and actuators.

With the goal of promoting this relation between customers and shop owners, the Smart Shopping experiments held in Santander are carried out in one of the most traditional markets in the city centre. The Connected Shop experiment [13] consists on the deployment of multiple devices to gather 802.11 probe request packets, used later to locate people in indoor scenarios. Results will be later given to the market managers to understand the interests of visitors and actuate accordingly. Moreover, the Advertised Premium Discounts experiment uses the same infrastructure to send offers to visitors. This is done using a Bluetooth interface, while the offers will be sent depending on the location of them, and taking this data from the Open Data platform of Santander.

2.2.4 Smart City

As an important part of the resources offered through the platform, the Smart City domain is related to urban areas that want to open collected data to citizen in order to ease the optimization of public services such as urban transportation, streetlights, water supply, waste recycling, etc. Hence, the federation provides large datasets and real-time data to research and develop new services and applications in the cities.

There are several applications based on Smart City data from which we can highlight the Smart Route Recommendation application. The goal is to provide the best route for tourists based on their preferences. For instance, if the tourist visits the city by car, the system will recommend him/her the best route to find a free parking spot in the city, while passing by important city sightseeing spots or the shops he/she might be interested.

3. FESTIVAL Architecture Design

FESTIVAL architecture aims at defining a general approach and methodology to be used to federate different testbeds and to use resources they provide through a unique point of access in order to reduce the complexity of discovery and of use.

FESTIVAL architecture specifies a common resource data model, the FESTIVAL Data Model, defined to provide the basic structure for shaping and representing the fundamental resource information such as description, capabilities, properties, etc. It also specifies two set of APIs; a set, EaaS APIs, is used by the experimenters to build and deploy rapidly and efficiently their experiments, whereas another set, Driver APIs, is used to enable FESTIVAL platform to communicate in a unique way with the different aggregators: the modules of the architecture responsible for aggregating testbeds providing the same type of resources, such as Open Data, IoT devices, IT resources and Living Lab resources.

FESTIVAL architecture is composed of four main layers (**Fig. 2**). The lowest one is the Uniform Access Layer; it aims at unifying the access to the four type of resources managed by FESTIVAL platform: Open Data, IoT devices, IT resources and Living Lab resources. Resources are hosted by different testbeds

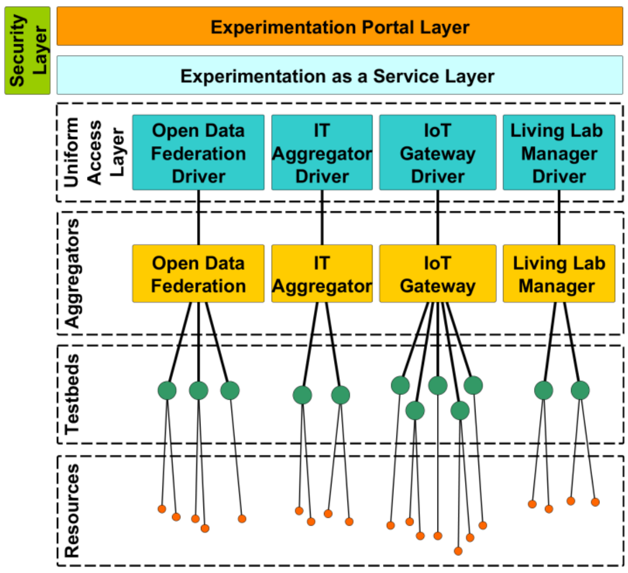


Fig. 2 FESTIVAL architecture.

and testbeds are aggregated by four "aggregators" where an aggregator exists for each type of resource. Each aggregator is attended by a driver implementing the Driver APIs providing resources following the FESTIVAL Data Model. The four drivers realize the Uniform Access Layer through the implementation of Driver APIs and the use of FESTIVAL Data Model to represent resources.

Experimentation as a Service Layer aims at providing the functionalities of the FESTIVAL platform to perform new experiments through EaaS APIs, a set of RESTful APIs designed to manage experiments and to discover resources available through the Uniform Access Layer.

Experimentation Portal Layer provides a unique point of access to the platform and to the federated set of FESTIVAL resources; this portal will give the experimenters the possibilities of creating a new account, accessing to the information of the different resources, and performing experiments, including the creation and management of experiments.

Finally, Security Layer provides an end-to-end secure access and control of the FESTIVAL resources through Experimentation as a Service Layer. Each request against Experimentation as a Service Layer is validated and authorized by this layer.

3.1 Uniform Access Layer

Uniform Access Layer is made by four drivers, one for each aggregator, implementing Driver APIs and providing description of available resources following FESTIVAL Data Model; the section of the data model related to resources is depicted in **Fig. 3**.

FESTIVAL platform provides specification of Driver APIs which are RESTful APIs and FESTIVAL Data Model to represent the different types of resources, Open Data resources, IoT devices, IT resources and Living Lab resources, in a uniform way. FESTIVAL Data Model provides the basic structure to shape and represent fundamental information of resources and data that play a key role in FESTIVAL.

Because FESTIVAL aims at providing a uniform access and representation of different types of resources, the process to define the FESTIVAL Data Model started from the analysis of the four type of resources managed by the platform, in order to identify their characteristics. This analysis built the basis to classify the characteristics that are in common to the four types of resources and that establish the main attributes to represent resources in FESTIVAL platform. Furthermore, in order to preserve peculiarities of each type of resources, FESTIVAL Data Model provides the possibility to represent them through a specific field: *additionalInfo*.

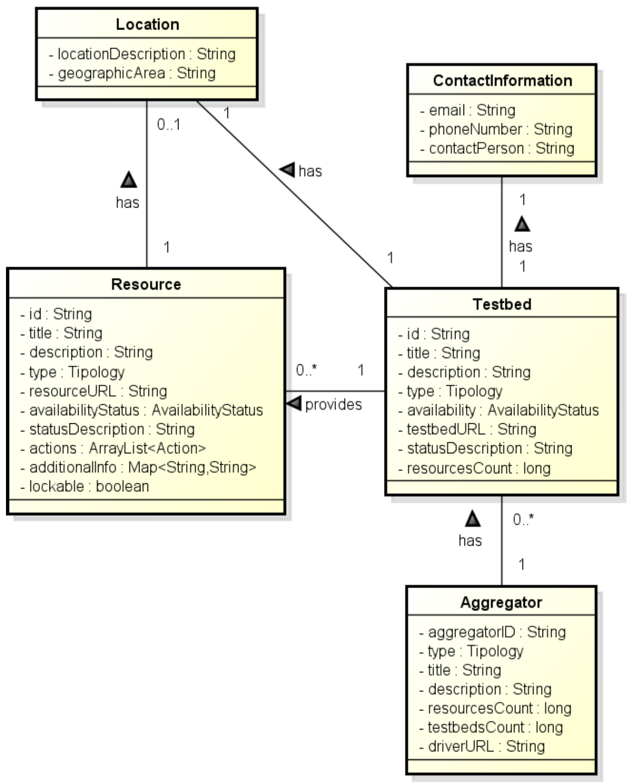


Fig. 3 FESTIVAL Data Model related to resources.

The central entity of the data model is entity "*Resource*" representing the resources provided by testbeds federated in FESTIVAL though their relative aggregator; apart general information, such as title and description of a resource, this entity also provides information about the type of a resource e.g., Open Data, IT Resource, via *type* field, the URL at which it is possible to retrieve more information about the resource or it is possible to interact with it via *resourceURL* field, its availability via *availabilityStatus* field, the reason why the resource is not available via *statusDescription* field, a list of possible actions executed on the resource via *actions* field, a list of additional information that can provide more details via *additionalInfo* field and the field *lockable* indicating that the resource which can be locked for an exclusive use for a certain period of time. A resource can have a location specifying its geographic position, that is provided by "*Location*" entity though a textual description *locationDescription* and *geographicArea*, a field providing a GeoJSON representing the location of the resource.

In order to contextualize resources, the data model provides an entity to represent testbeds owning them, through the relation "*Testbed provides Resource*"; entity "*Testbed*" is mainly characterized by a title, a description and a type. Moreover, it is possible to provide an URL at which it is possible to retrieve more information about the testbed via *testbedURL* field, its availability via *availabilityStatus* field, the reason why the resource is not available via *statusDescription* field, and the count of available resources on the testbed via *resourcesCount* field. The same as a resource, a testbed can have a location specifying its geographic position. Furthermore, a testbed can have contact information that is provided by entity "*ContactInformation*" through an email address via *email* field, a phone number via *phoneNumber* field and the name of a contact person via *contactPerson* field.

Finally, testbeds are aggregated by aggregators through the relation "*Aggregator has Testbed*." Aggregators manage testbeds offering resources of the same type and are represented by the entity "*Aggregator*"; entity "*Aggregator*" is characterized by a title, a description, a type, by an URL representing the end point of the driver provided by the aggregator implementing the Drive APIs via *driverURL* field. This URL is used by internal modules of Experimentation as a Service Layer in order to interact with the drivers and the aggregators. Moreover, the two fields *resourcesCount* and *testbedsCount* provide respectively the total number of resources provided by the aggregator as sum of the resources provided by the testbeds it aggregates and the total number of testbeds it provides.

It is important to underline that FETSIVAL platform does not assign any ID to resources and to testbeds, but only to the aggregators. IDs to resources and to testbeds are assigned by aggregator in order to obtain a more flexible and agile management of resources; so, each resource is uniquely identified not by its ID, but though the combination of three IDs: aggregator ID, testbed ID and the resource ID.

As described in the previous sections, FESTIVAL provides four aggregators to aggregate testbed managing the four types of resources:

Open Data Federation for Open Data testbeds, IoT Gateway (sensiNact [14]) for IoT testbeds, FESTIVAL IT Aggregator for testbeds of IT resources and Living Lab Manager for Living Labs.

The details of each aggregator are as follows.

- Open Data Federation aggregates Open Data Portals providing their resources, in terms of standard open data such as csv, xml, pdf and so on, or Linked Open Data, e.g., RDF, to experimenters in the Festival Federation; at the moment it aggregates the following Open Data Portals: Santander Datos Abiertos, providing open data about Municipality of Santander, Grand Lyon, providing open data about Municipality of Lyon, FIWARE Lab Data Portal, providing open data from different cities and regions in Europe and FESTIVAL Japanese Open Data Platform, providing information about experiments and applications deployed in Japan.
- IoT Gateway (sensiNact) aggregates different testbeds providing IoT devices: Connectivity Technologies Platform PTL, providing an environment to test connectivity technologies under realistic conditions, iHouse, providing provides an experimental smart house facility, ATR Data Center, providing an experimental data center facility, SmartSantander which is an experimental facility to test solutions and technologies in the context of a city.
- FESTIVAL IT Aggregator aggregates two IT testbeds providing computational resources such as Virtual Machines: JOSE, providing template of Virtual Machines with different amount of memory and power computation and FIWARE Lab, providing template of Virtual Machines with different amount of memory, power computation and preinstalled FIWARE Generic Enablers.
- Living Lab Manager aggregates two Living Labs providing services, locations, expertise, methodologies and human resources, such as communities involved in the activities of Living Labs: TUBA, located in Lyon – France and Knowledge Capital The Lab. located in Osaka – Japan.

Each aggregator implements the Driver APIs and expose a driver reachable from Experimentation as a Server Layer of the FESTIVAL architecture. Each implementation of the Driver APIs is independent and free to adopt any technology and to use any solution to provide access to resources. Indeed, the only constrain is to respect specification of Driver APIs and FESTIVAL Data Model.

Driver APIs provide abstract and generic methods to access and use resources; Each aggregator is responsible to implement these APIs and to adapt them for its specific resources. Two emblematic examples are Living Lab Manager Driver and FESTIVAL IT Aggregator Driver. About the former, because the nature of the resources provided by Living Labs that are mainly human resources and their expertise, it is not possible to manage them automatically. A direct contact between an experimenter and the Living Labs is required if the experimenter plans their involvement in his/her experiments. On the opposite side, FESTIVAL IT Aggregator Driver automates completely the access to IT resources, i.e. Virtual Machines, for instantiating them and providing a direct access to experimenters.

3.2 EaaS Layer

Experimentation as a Service layer, or EaaS, is one of the central pieces of FESTIVAL architecture duo to its evolutional capability and its efficient communication with other components of the architecture. The former reason can be justified by the efficiency in which the new functionalities can be created, since this layer can be extended during the runtime as J2EE component. This characteristic allows the evolution of the central piece without tearing apart the already built components. The latter reason is that the communication among components at this layer can be very efficient, since those components share the same runtime environment, though HTTP is used in the case for intercommunication, the other communication channels can be used to exchange information among those components, which is an important property when dealing heavy processing requirements, which is the case for future Data Analytics.

EaaS logical deployment in the architecture is equidistant-situated from the other components of this architecture making of it an inherent optimized component to deploy the business intelligence.

EaaS is transversal with the respect to the silos tacked by the platform – OpenData, IoT, IT and LivingLab resources. This transversal property implies that this layer has a large

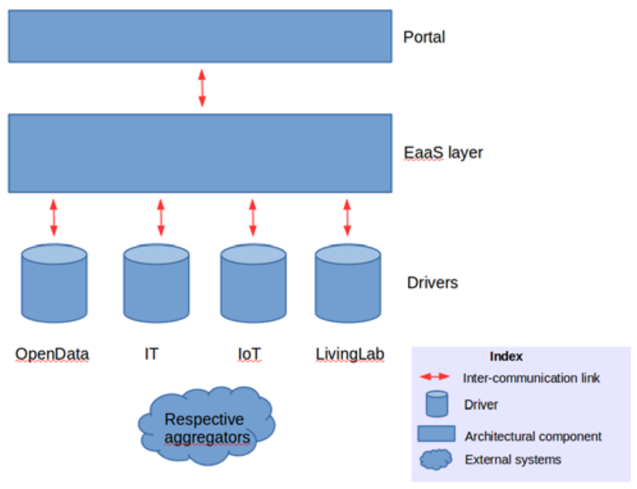


Fig. 4 Inter-communication link.

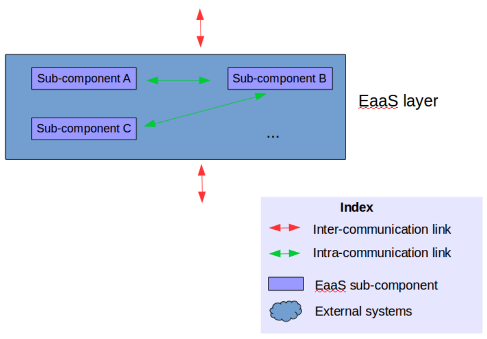


Fig. 5 Intra-communication link.

accessibility span to other parts of FESTIVAL architecture. EaaS has mainly two different types of communication: Intra-communication (**Fig. 4**) and Inter-communication (**Fig. 5**).

The intra-communication is the communication established among components that area placed inside the EaaS itself as a sub-component, meaning that a component A and B, both deployed in EaaS, can establish a direct can communication. This communication can be uni-directional or bi-directional, depending on the requirement of this exchange. This type of communication can be highly optimized, several techniques Inter-process communication (IPC) can be used.

The inter-communication is the communication of one EaaS component with another FESTIVAL architecture component, usually this communication follows Representational State Transfer (RESTful) style, which is implemented via HTTP protocol, but it's not exclusively.

One third type of communication exists, but is not mentioned in this document, it is the communication used between the Driver and the Aggregator itself, since this communication is mandatory, the FESTIVAL architecture cannot have influence on it, thus it's not a direct part of the FESTIVAL architecture.

3.3 Security Layer

The security in FESTIVAL has been implemented following the OAUTH2 protocol [15] as a separate layer from the platform. The architecture has been implemented using three Generic Enablers from FIWARE, as described below:

- Identity Management Keyrock (IDM) [16]: this is the core

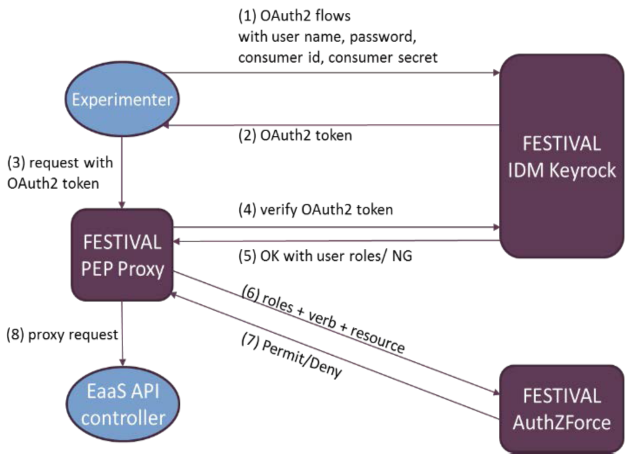


Fig. 6 Security Module API request flow.

component of the security module. This Generic Enabler is in charge of managing the experimenter accounts, as well as their roles, in the FESTIVAL platform. Additionally, this component creates and manages limited-time tokens that can be used to authenticate and authorise an experimenter into the platform. The IDM also provides single sign-on functionality, enabling future federation with other platform or services.
- PDP AuthZforce proxy [17]: the PDP (Policy Decision Point) proxy is the component that is in charge of the authorisation process of the security module. This component provides the authorisation mechanism to the EaaS platform depending on the experimenter roles. It uses a set of XACML [18] files that link the experimenter roles with the corresponding rights.
- WILMA PEP proxy [19]: the PEP (Policy Enforcement Point) proxy is a middleware which acts as a firewall and will authenticate and authorise the user accessing to the platform with the token against the IDM and the PDP components respectively. Hence, each call made against the platform will be authenticated and authorised using the PEP proxy.

The security access flow in FESTIVAL is depicted in **Fig. 6**. So as to access to the FESTIVAL EaaS APIs, the user will need to perform a request for a token to the FESTIVAL Keyrock instance using his/her credentials. This token will be later used to access to the EaaS platform, where the PEP proxy checks for the authentication and authorisation. If the user has the appropriate rights, the request will be forwarded to the EaaS API Controller.

3.4 Portal Layer

In the portal layer, EaaS components are graphically integrated and provided to the experimenters. An example experimentation flow via portal is described as follows:

- An experimenter accesses to the portal for starting an experiment.
- Security layer interrupt the request and authenticate/ authorise the experimenter. When the experimenter successfully login to the platform, the request is redirected to the EaaS layer.
- The experimenter creates a new experiment and reserves re-

sources via the portal. EaaS layer handles requests from the portal via EaaS APIs as described in Section 3.2.

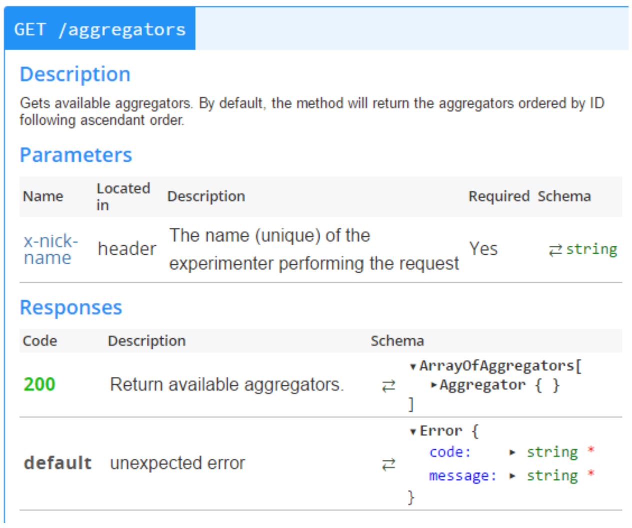
- EaaS layer interacts with testbeds via aggregator and driver implementations to reserve/release resources.
- The obtained results or status of the request are returned to the experimenter by visualization functions of the portal.
- After expiration of the experiment period, the experimenter renews the experiment or release the resources related to the experiment.

As described in Section 3.4, security layer can handle the both direct access from experimenters and indirect access via the portal. Experimenters can choose a proper access method from GUI base manual access and programmable API base access.

4. Functional Tests and Validation

EaaS Layer is the highest level of integration of all the FESTIVAL components. A set of APIs has been defined respectively for both inter-communication links involving the EaaS Layer, following a RESTful style. Due to page limitation, we cannot list all the APIs here. The following **Fig. 7** gives an example of the API "GetAggregator" provided by the EaaS Layer to Web portal intercommunication link. Functional test on this set of APIs (EaaS APIs) can prove a correct integration of all the FESTIVAL components since the desired behavior of higher-level components relies on the correct integration of lower-level components as depicted in Fig. 2.

The black-box MBT (Model Based Testing) [20] approach is applied for the EaaS platform level test development. This approach considers a sub set of UML models, composed from class and object diagrams. The class diagram describes the system's structure, e.g., the entities with parameters and the operations representing the API functions. The modelling of the FESTIVAL platform is shown in **Fig. 8** (the "sut" class, partially). The object diagram instantiates the class diagram with the test input data. The behavior of the system is described by OCL (Object Constraint Language) constraints written as pre/postcondition of each operation. **Figure 9** shows the expected behavior of the operation "GetAggregator" regarding the API specification. This model is



GET /aggregators

Description

Gets available aggregators. By default, the method will return the aggregators ordered by ID following ascendant order.

Parameters

Name	Located in	Description	Required	Schema
x-nick-name	header	The name (unique) of the experimenter performing the request	Yes	⇄ string

Responses

Code	Description	Schema
200	Return available aggregators.	⇄ ▾ArrayOfAggregators[▸Aggregator { }]
default	unexpected error	⇄ ▾Error { code: ▸ string * message: ▸ string * }

Fig. 7 EaaS API example: "GetAggregator" API.

then passed to a test generation software to produce test plan, test cases and the "requirements to test cases traceability matrix" that is showing the relationship between requirements and test cases using the tag "@REQ" that we can observe in Fig. 9. The generated tests need to be adapted with concrete value of the test target, for example, the URL of the EaaS platform endpoint, in order to be executed on the EaaS platform and test results can be easily displayed with a "pass/fail" status because the generated test cases include verification points that are tagged with "@AIM" in Fig. 9.

The most significant advantages using MBT are: 1) Automatic generation of test cases from the system model; 2) Traceability of requirements to know which requirement is covered by which test.

The tests have been performed on the current EaaS Platform instance which is still under development. Among 46 generated test cases executed, 32 passes and 14 failed. The 32 success showed us the core API functions have been correctly implemented and integrated, for example, the resource discovery, the experiment creation. For the 14 test cases that failed, there are three main reasons: missing error code list in the specification (3 test cases, an extract of report is shown in **Fig. 10**), unimplemented APIs (4 test cases) and bad interpretation of specification (7 test cases). Figure 10 shows an example of detected error. In this specific case, the expected error code was not specified in the specification. The tester assumed the code was 404 (in the first red rectangle) and waited for it, whereas the developer implemented code 200 (in the second red rectangle) which is different from the tester's expectation. The solution to this error case is to specify explicitly the error code in the specification in order to make the expected code and implemented code the same. As the implementation and integration is still ongoing, the unimplemented

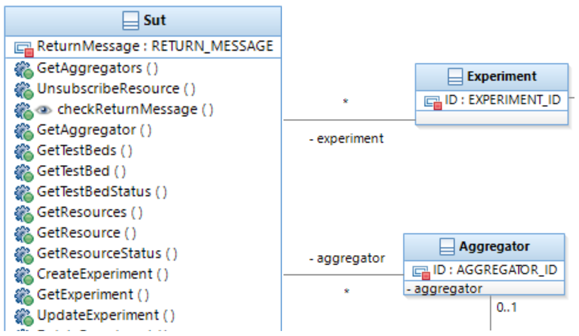
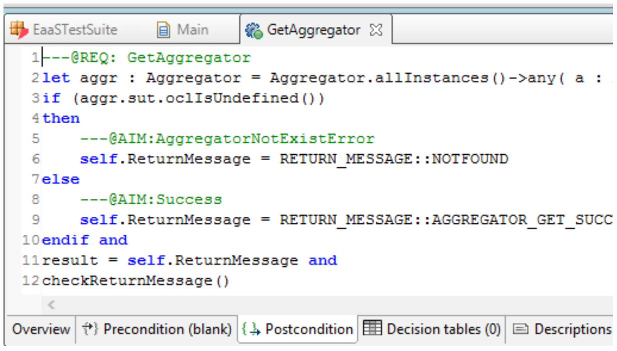
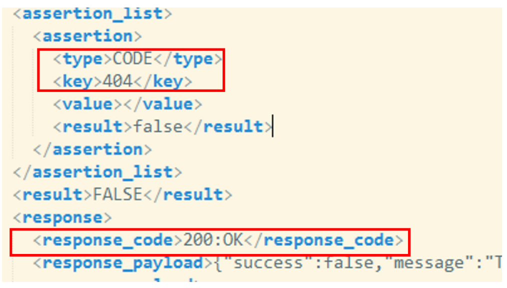


Fig. 8 EaaS Platform MBT model-class diagram (partial).



```
---@REQ: GetAggregator
let aggr : Aggregator = Aggregator.allInstances()->any( a :
if (aggr.sut.oclIsUndefined())
then
    ---@AIM:AggregatorNotExistError
    self.ReturnMessage = RETURN_MESSAGE::NOTFOUND
else
    ---@AIM:Success
    self.ReturnMessage = RETURN_MESSAGE::AGGREGATOR_GET_SUCC
endif and
result = self.ReturnMessage and
checkReturnMessage()
```

Fig. 9 EaaS Platform behaviour of function "GetAggregator."



```
<assertion_list>
  <assertion>
    <type>CODE</type>
    <key>404</key>
    <value></value>
    <result>false</result>
  </assertion>
</assertion_list>
<result>FALSE</result>
<response>
  <response_code>200:OK</response_code>
  <response_payload>{"success":false,"message":"T
```

Fig. 10 Example failed test case due to not specified error code.

APIs will be made available soon, the specifications will be updated with the error code list and clearer definition of some points that created confusion in interpretation will be made. An interoperability event will be organized to test the final EaaS Platform when the implementation and integration is finalized.

5. Consideration

The first implementation of FESTIVAL platform appropriately federates existing testbeds to provide required functions for the investigated use cases as EaaS APIs. It assumes that experimenters develop their own application to utilize resources via EaaS APIs. However, an issue is remaining for experimenters to execute their experiments. **Figure 11** (a) depicts the issue in developing experimenter's application. EaaS API abstracts existing testbeds and makes them easier to integrate. However, experimenters are still required to integrate those APIs in their applications. Furthermore, while FESTIVAL platform provides intercontinental testbeds to the experimenters, it still lacks geographical scalability. Sometimes, experimenters require not only FESTIVAL APIs but also external platform APIs to obtain additional data and functions. Since a certain number of experimenters are focus on analyzing data and not interested in developing applications, the integration cost cannot be ignored. Figure 11 (b) shows a candidate solution for the issue where *application templates* for integrating EaaS APIs and external platform APIs are provided. Based on the template, experimenters can develop their applications easier. In the following, we will explain an approach to provide application templates.

Since FESTIVAL platform has IT Aggregator and provides IT resources as virtual machine instances, application templates can be provided as virtual machine images. However, a suitable environment for experimenters varies widely depending on their requirements. In order to provide various templates for them, flexible configurability is required for the platform provider. IT automation tools, e.g., Chef [21], Ansible [22], can be a solution to the problem. In Chef case, know-hows of setting up and federating middleware can be provided as a set of scripts, named cookbook. FESTIVAL platform provides several cookbooks via software repositories for setting up environments considering testbed characteristics. Since each application domain has typical patterns of data collection and analysis procedure, not only communication but also data analysis know-hows can be provided as application templates. For example, time series database, e.g., Elasticsearch, and visualization tool, e.g., Kibana, can be easily deployed into the VMs as a basic analysis tool.

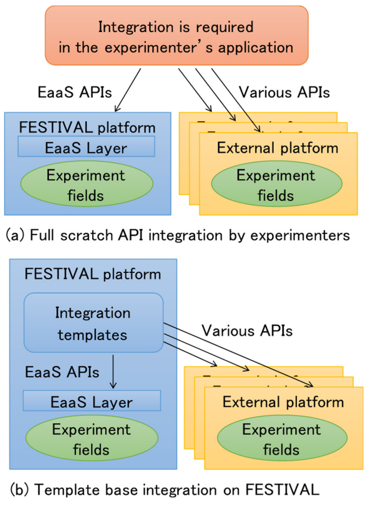


Fig. 11 Issues in developing experimenter's application.

Furthermore, external platform APIs provide their data and functions with various communication styles, e.g., asynchronous, synchronous, push and pull. A communication middleware bridging various input and output APIs for end-user level integration is required. In FESTIVAL platform, sensiNact is adopted as a platform level integration middleware. For end-user level, another integration middleware fluentd [23] is also provided. While fluentd is basically designed for collecting log data from servers, it can be used for IoT applications because it has various input and output plugin to connect messaging and database middleware. FESTIVAL project provides MQTT input/output plugins and external platform connector for EverySense service. EverySense [24] provides data exchange services mediating between sensor data providers and sensor data requesters. Sensor data providers can set the terms and conditions about who can use the data, how to use the data, and the price of the data. Sensor data will be anonymized, cleared from any personal identifiable information before delivering to the companies who want to use the data for their own purposes. Since it enhances geographical scalability of sensor data and brings data with providers' approval, experimenters can easily extend their experiment area. It also enables experimenters to distribute their analysis results into the data exchange market. This kind of external platform federation delivers more values to FESTIVAL platform. Further investigation must be required to establish more flexible and valuable platform.

6. Conclusion and Future Work

In this paper, FESTIVAL platform design and its implementation was introduced. Since the platform design was based on the investigation of the testbeds and target applications, testbeds utilized in the current implementation and representative use cases were introduced in Section 2. The platform architecture and overview of its implementation were also described in Section 3. The implementation is under validation process of model based testing as described in Section 4. Furthermore, a consideration of remaining issues was discussed in Section 5. The platform still doesn't reach the final shape but get into the phase of evaluation with external experimenters. The feedbacks from the experimenters are expected to improve the platform usability and applicability not only in the current applications but also in the extended application domains.

The FESTIVAL architecture has proven to be reliable and capable in connecting remote experimentation sites, which answers part of its main purpose: to serve as key tool for Smart ICT experimenters.

Currently the FESTIVAL platform is undergoing on an internal evaluation process to detect future problems that will be tacked by the platform. The preliminary problematics brought to the table by the project stakeholders point to data maintenance, distribution and analysis. Three branches there have been exploited and engendered internal working groups to establish initial solutions and research entities to be involved.

Acknowledgments A part of the study was carried out by the FESTIVAL project, the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN.

References

[1] Lyon, G.: Lyon Open Data Portal, available from ⟨http://data.grandlyon.com/⟩ (accessed 2016-12-12).
[2] Sanchez, L. et al.: SmartSantander: IoT experimentation over a smart city testbed, *Computer Networks*, No.61, pp.217–238 (2014).
[3] Tan, Y.: Smart Home and Smart Community Simulator, *6th Japan-EU Symposium on ICT Research and Innovation* (2016), available from ⟨http://www.soumu.go.jp/main_content/000445949.pdf⟩.
[4] Enkhee, T., Hasegawa, G., Tarutani, Y., Matsuda, K. and Matsuoka, M.: Large-scale ASP-based HEMS Utilizing Interactive Web Technologies, *IEEE SmartGridComm* (2015). available from ⟨http://ieeexplore.ieee.org/document/7436399/⟩.
[5] IRT PULSE, available from ⟨http://www.irtnanoelec.fr/pulse/⟩ (accessed 2016-12-12).
[6] Nishinaga, N. and Harai, H.: Progress and Results of New-Generation Network Research and Development, *Journal of NICT*, Vol.62, No.2 (2016), available from ⟨http://www.nict.go.jp/publication/shuppan/kihou-journal/journal-vol62no2/journal-vol62no2-03-00.pdf⟩.
[7] TUBA, available from ⟨http://www.tuba-lyon.com/⟩ (accessed 2016-12-12).
[8] The Lab., available from ⟨http://kc-i.jp/en/facilities/the-lab/⟩ (accessed 2016-12-12).
[9] Tarutani, Y. et al.: Reducing Power Consumption in Data Center by Predicting Temperature Distribution and Air Conditioner Efficiency with Machine Learning, *2016 IEEE International Conference on Cloud Engineering* (*IC2E*), Berlin, pp.226–227 (online), DOI: 10.1109/IC2E.2016.39 (2016), available from ⟨http://ieeexplore.ieee.org/document/7484193/⟩.
[10] Deguchi, T., Taniguchi, Y., Hasegawa, G., Nakamura, Y., Ukita, N., Matsuda, K. and Matsuoka, M.: A Workload Assignment Policy for Reducing Power Consumption in Software-Defined Data Center Infrastructure, *IECE Trans. Comm.*, Vol.E99.B, No.2, pp.347–355 (online), DOI: 10.1587/transcom.2015EBP3267 (2016).
[11] Tsuchiya, K., Nishida, J., Yoshida, R. and Shirahama, S.: A consideration on the improvement of information providing in stations using ICT technology, *53rd Conference of the Committee of Infrastructure Planning and Management* (2016).
[12] FIWARE, available from ⟨https://www.fiware.org/⟩ (accessed 2016-12-12).
[13] Santana, J.R. et al.: A low-cost solution for tracking visitors in Smart Shopping environments: A real platform implementation based on Raspberry Pi, *Proc. Monami* (2016).
[14] sensiNact, available from ⟨http://organicity.eu/tools/sensinact/⟩ (accessed 2016-12-12).
[15] OAUTH2 protocol, available from ⟨https://oauth.net/2/⟩ (accessed 2016-12-12).
[16] Identity Management – KeyRock, available from ⟨https://catalogue.fiware.org/enablers/identity-management-keyrock⟩ (accessed 2016-12-12).
[17] Authorization PDP – AuthZForce, available from ⟨https://catalogue.fiware.org/enablers/authorization-pdp-authzforce⟩ (accessed 2016-12-12).
[18] XACML, available from ⟨http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf⟩ (accessed 2016-12-12).
[19] PEP Proxy – Wilma, available from ⟨https://catalogue.fiware.org/enablers/pep-proxy-wilma⟩ (accessed 2016-12-12).
[20] Utting, M., Pretschner, A. and Legeard, B.: A taxonomy of model-based testing approaches. Software Testing, *Verification and Reliability*, Vol.22, No.5, pp.297–312 (2012).
[21] Chef, available from ⟨https://www.chef.io/⟩ (accessed 2016-12-12).
[22] Ansible, available from ⟨https://www.ansible.com/⟩ (accessed 2016-12-12).
[23] fluentd, available from ⟨http://www.fluentd.org/⟩ (accessed 2016-12-12).
[24] EverySense, available from ⟨http://every-sense.com/en/⟩ (accessed 2016-12-12).

Toyokazu Akiyama received his B.E., M.E. and Ph.D. degrees in Information Systems Engineering from Osaka University, Japan in 1997, 1999 and 2003, respectively. Since 2000, he worked as a Research Associate (Assistant Professor) at the Cybermedia Center, Osaka University. Currently, he is an Associate Professor of the Faculty of Computer Science and Engineering, Kyoto Sangyo University. His research interests include distributed systems and internet applications. He is a member of the IEEE Computer Society, IPSJ, IEICE.

Shuuichirou Murata received his B.S. and M.S. in Engineering at Kyushu Institute of Technology, Japan in 1996 and 1998 respectively. Since 1998, he worked at ASTEC, Inc. Currently, he worked at Acutus Software, Inc.

Kiichi Tsuchiya received his B.E. in Economics from Keio University, Japan in 1993. Since 1993, he worked at West Japan Railway Company. Currently, he worked at JR West Japan Communications Company.

Teruaki Yokoyama received his B.E. in Computer Science from Ritsumeikan University, Japan, in 2000, and his M.E. and Ph.D. degrees in Information Science from the Nara Institute of Science and Technology, Japan in 2002 and 2007. His research interests include platforms and applications for networked system in developing countries. He is a member of the Widely Integrated Distributed Environment (WIDE) Project and Asian Internet Infrastructure Initiatives (AI3). He is currently a Lecturer at the Kobe Institute of Computing, Japan.

Martino Maggio is Senior Researcher and Project Manager. He got his Computer Science Engineering Master Degree in July 2005 from University of Palermo and in the same year he started to work as a researcher in Engineering Ingegneria Informatica R&D laboratory. He has been involved in several European and Italian research projects as technical manager and work package coordinator. During the last years he has contributed in writing many proposals for EU co-funded research projects, in programs such as the FP7, the CIP PSP ICT and H2020. At present he is involved in two H2020 EU-Japan collaborative projects: BigClouT in the domain of Cloud Computing and FESTIVAL to implement experimental testbed federation. PMP PMI Certified in 2014.

Giuseppe Ciulla got his Master Degree in Computer Science Engineering from University of Palermo in April 2006. After a first experience at CNR – Italian National Research Council, he started to work at R&D Laboratory of Engineering Ingegneria Informatica in 2009. In Engineering he has been involved in several Italian and European research projects carring out different technical activities related to software design and development. Currently he is involved in EU-Japan FESTIVAL project (H2020).

Juan Ramón Santana is a Telecommunication Engineer graduated in 2010 in the University of Cantabria. He is currently working as research fellow in the Network Planning and Mobile Communications Laboratory, a telecommunication research group from the same university. Prior to his current occupation, he was also part of the University of Strathclyde (Glasgow), working on IoT solutions for the cattle industry. He has been involved in several projects, such as SmartSantander or EAR-IT, European collaborative projects related to the Smart City paradigm and the Internet of Things. Among his research interests are WSN (Wireless Sensor Networks), M2M communications and smartphone applications.

Mengxuan Zhao has a Ph.D. in Computer Science from University of Grenoble. With a strong background and research interest in the IoT domain and the semantic technologies, she mainly works on European research projects including Fiesta-IoT (semantic interoperability of testbeds), Festival (EU-JP, interoperability and federation of ICT services and testbeds). She also participates and contributes to standardization works, including participation and organization of plugtest events.

Jander Botelho do Nascimento is a Researcher Engineer at CEA, obtained his Software Engineer diploma in Brazil. He worked as Project Leader/Architect in several industrial software for large-scale enterprise platforms. Later on acquiring the diploma of Masters in distributed systems and applied mathematics in the University of Grenoble Alps – France. Nowadays he emphasis his work in Internet of Things domain and distributed architecture with edge technologies.

Levent Gürgen is R&D project manager in CEA-LETI and involved in several collaborative R&D projects about IoT and smart cities. He is in particular coordinating 2 European-Japanese collaborative projects, namely BigClouT and FESTIVAL. Levent obtained his PhD degree in computer science from the Grenoble Institute of Technology in 2007. His main research interests include sensor data processing and service-oriented platforms for Internet of Things.