Advancing Experimentation-as-a-Service Through Urban IoT Experiments

Dimitrios Amaxilatis, Dennis Boldt, Johnny Choque, Luis Diez, Etienne Gandrille, Sokratis Kartakis, Georgios Mylonas, and Lasse Steenbock Vestergaard

***Abstract*—Smart cities are becoming a vibrant application domain for a number of science fields. As such, service providers and stakeholders are beginning to integrate co-creation aspects into current implementations to shape the future smart city solutions. In this context, holistic solutions are required to test such aspects in real city-scale Internet of Things (IoT) deployments, considering the complex city ecosystems. In this paper, we discuss OrganiCity's implementation of an experimentation-as-a-service (EaaS) framework, presenting a toolset that allows developing, deploying, and evaluating smart city solutions in a one-stop shop manner. This is the first time such an integrated toolset is offered in the context of a large-scale IoT infrastructure, which spans across multiple European cities. We discuss the design and implementation of the toolset, presenting our view on what EaaS should provide, and how it is implemented. We present initial feedback from 25 experimenter teams that have utilized this toolset in the OrganiCity project, along with a discussion on two detailed actual use cases to validate our approach. Learnings from all experiments are discussed as well as architectural considerations for platform scaling. Our feedback from experimenters indicates that EaaS is a viable and useful approach.**

***Index Terms*—Evaluation, experimentation, Internet of Things (IoT), real-world deployment, smart cities.**

I. Introduction

INFORMATION and communication technologies (ICTs) are being adopted as a catalyst in the smart city domain, on top of which novel services are developed and legacy ones are evolved. In addition, the pervasive presence of ICT allows service providers and stakeholders to directly interact with citizens to continuously improve services in a co-creative manner. This co-creation approach permits the rapid deployment and adoption of innovative solutions for urban challenges [1]. A central part of instrumenting a smart-city with an Internet of Things (IoT) infrastructure is to be able to sense and monitor how the city performs, and to understand the dynamics of the city as a system. As examples, it becomes possible to examine how people make use of urban environments, and parameters like pollution, can be visualized and tracked throughout the city.

Recently, co-creation has been exploited in different ways by various experimentation-as-a-service (EaaS) frameworks [2]. By "co-creation," we refer to the process of involving citizens and other stakeholders in designing as well as developing smart city solutions, and using local know-how to respond to existing challenges in modern cities and their communities. For instance, Pallot and Pawar investigated co-creation to improve QoS [3], while gamification techniques are used by Pokrić *et al.* [4]. Participation and interaction with stakeholders are used to attract people to the co-creation process [5]–[7]. Finally, other work, such as Schaffers *et al.* [8], focuses on technical aspects, in particular IoT or machine-to-machine communications, leaving aside user/citizen interaction. In this sense, the scope of existing EaaS frameworks is limited to specific services or challenges. However, we believe that end-to-end solutions are necessary for harnessing the potential of complex city ecosystems. At the same time, the diversity in potential smart city applications makes it hard for a monolithic solution to satisfy their varied requirements.

In this context, the OrganiCity[1] EU project (OC) developed a customizable EaaS framework, called the *OC platform*. The OC platform provides a common playground where different stakeholders can co-create urban services [9], and adopt the functionalities offered by the platform that better serve their needs. As depicted in Fig. 1, its design follows a three-tier approach [10], addressing data provisioning, platform management, and experimentation support. It has been developed by following an experimentally driven approach, supported by *open calls* during two periods (first open call in 2016–2017 and second open call in 2017–2018). Open calls have had a twofold objective: 1) to help maturing the platform by gathering feedback from external users and 2) to analyze the platform sustainability for the future. Throughout this paper, accepted open call projects are termed *experiments*, and the actual people conducting these are referred to as *experimenters*

Manuscript received June 20, 2018; revised August 28, 2018; accepted September 16, 2018. Date of publication September 24, 2018; date of current version May 8, 2019. This work was supported by the OrganiCity Research Project funded by the European Union of the Horizon 2020 Research and Innovation Program under Grant 645198. *(Corresponding author: Luis Diez.)*

D. Amaxilatis and G. Mylonas are with the Computer Technology Institute and Press Diophantus, 26500 Rio, Greece (e-mail: amaxilat@cti.gr; mylonasg@cti.gr).

D. Boldt is with the Institute of Telematics, University of Lübeck, 23562 Lübeck, Germany (e-mail: boldt@itm.uni-luebeck.de).

J. Choque and L. Diez are with the Communications Engineering Department, University of Cantabria, 39005 Santander, Spain (e-mail: jchoque@tlmat.unican.es; ldiez@tlmat.unican.es).

E. Gandrille is with the Commissariat à l'énergie atomique et aux énergies alternatives, 38054 Paris, France (e-mail: etienne.gandrille@cea.fr).

S. Kartakis is with Intel Labs Europe, London E14 5NR, U.K. (e-mail: sokratis.kartakis@intel.com).

L. S. Vestergaard is with Alexandra Instituttet, 8200 Aarhus, Denmark (e-mail: lasse.vestergaard@alexandra.dk).

Digital Object Identifier 10.1109/JIOT.2018.2871766

[1][Online]. Available: OrganiCity H2020 project, http://organicity.eu

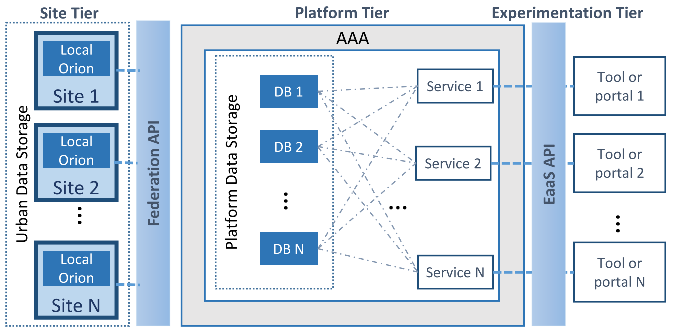


Fig. 1. Overall OrganiCity platform architecture.

and *experiment team* interchangeably. In addition, the term *experimentation* refers to the process of conducting an actual experiment.

The first version of the OC platform [9] was designed and validated within the OrganiCity consortium. This initial version was exploited during the first open call, which served to gather information and feedback from experimenters. Afterward, the platform was tweaked according to the feedback and the new version was released in the second open call, which was devoted to developing a co-creation methodology for establishing cross-collaboration between stakeholders, communities, sectors, and countries in a smart city context. In this paper, we focus on the technical evolution, so that the work presented hereinafter spans from the period from which the OC platform was first deployed (initiation of first open call) to shortly before the second open call. In the first open call, a number of experiments have been selected and implemented, over a period of six months. Thanks to this key part of the OC project, experimenters have played an active role in the design and validation of our platform.

In the following, we present a toolset to foster the development of urban services in a co-creative way. These novel services have been identified based on the feedback provided by 25 experiments conducted in five European cities (being from U.K., Denmark, Spain, Greece, and Belgium), which have resulted in creating pilot services for more than ten different urban application areas. In addition, the experiment teams involved different types of stakeholders, ranging from individual citizens to small and medium-sized enterprises (SMEs), or public authorities. To our best knowledge, this is the first time that an EaaS framework is adopted in such a large scale manner, spanning over multiple cities.

The rest of this paper is structured as follows. First, Section II provides a discussion of related work, highlighting differences between existing approaches and ours. Then, In Section III, the most relevant feedback obtained during OC platform exploitation, as well as possible modifications are explained. In Section IV, we describe the services developed after the first experimentation period, highlighting their functionalities and benefits. Finally, Section V concludes this paper.

II. Related Work

In terms of experimentation testbeds, SmartSantander [11] built one of the largest city-scale IoT research infrastructures, pioneering the experimentation of novel smart city architectures, services, and applications in real-world urban environments. It emphasized managing experiments at an IoT device level, while allowing data-acquiring tasks facilitating urban services on top of the captured data flows. SmartSantander built interfaces with FIWARE and FI-Lab,[2] to support interconnectivity with the IoT/Future Internet community. WISEBED [12] pioneered discrete IoT device testbed federation.

Additionally, projects like IoT-Lab[3] investigate crowdsourcing and IoT services for supporting multidisciplinary research tasks. Their approach differs from ours, since their services are not tightly coupled with a smart city testbed. Festival [13] is another example of existing federating experimentation testbeds from Europe and Japan, in order to provide a unified infrastructure to the research community. However, its focus is not exclusively on the smart city domain, and it does not offer the scale or the toolset provided by OC as a platform overall. CPaaS.io [14] is an ongoing project with similar goals to OC. Synchronicity[4] features open calls for developing new services, similar to OC, but places a much larger focus on open data markets in the context of a smart city, leaving aside the co-creative approach.

In light of these advancements, OC aims to combine the aforementioned approaches and co-create new smart city solutions with citizens, researchers, and city authorities. Crowdsensing (i.e., tasking groups of volunteers to gather various kinds of data using IoT devices and/or smartphones) is one of the directions taken to address this challenge, helping to build a smart city data repository. Regarding the OC platform, a more general discussion is provided in [9], describing its architecture and overall software stack, with potential co-creation capabilities showcased in [15] in detail. Apart from aiming to provide a pragmatic solution to the crowdsensing problem, the toolset discussed in this paper allows end-users to benefit from this interoperability in various ways (data storage, visualization, interfacing to other systems, community management, knowledge extraction, and urban service creation), and not just basic management of crowdsensing activities.

As an example of this approach, OC's crowdsensing tool allows for a broad set of opportunities for integrating smartphones in a smart city experimentation context, allowing the use of such devices to produce experimental data as an extension of an existing IoT infrastructure. The mobile crowdsensing paradigm and the associated features and challenges are further discussed in [16]. With respect to incentives and crowdsensing task assignment, which are also part of this paper, in [17] and [18] such aspects are discussed in detail. In addition, [19] and [20] provides a discussion related to several research questions we are trying to answer in OC as well. Finally, as discussed in Section I, OC has managed to appeal to a large number of research teams so far, and has already produced a large number of urban experiments, far surpassing the respective numbers kick-started by other similar projects.

[2][Online]. Available: https://lab.fiware.org
[3][Online]. Available: http://www.iotlab.eu
[4][Online]. Available: http://synchronicity-iot.eu

III. Learnings

In this section, we discuss learnings from the initial deployment of the OC platform during the first open call. We shortly present two implemented experiments, and the experimentation teams' experiences from utilizing the OC platform. In continuation, we discuss the most common issues and comments experimenters have provided regarding their experiments. Finally, we discuss architectural considerations on how to robustly scale the OC platform in future deployments.

A. Experimenter Experiences

As mentioned in Section I, 25 experiments were conducted in five different European cities during the first open call. All experiments used the OC platform, and each experiment team was required to report on their learnings with the OC platform. These reports were conducted twice: first through an interim standardized questionnaire, and then again at the end of the experiment. In this paper, we put emphasis on those inputs provided in the final questionnaires, since they bring richer descriptions of experiences with the technical part of the OC platform, than the interim questionnaires. In the following sections, we present two specific experiments in order to show practical examples of how the OC platform has been utilized. We then discuss the most prevalent learnings reported by all experimenters.

1) Spend Network: This experiment[5] aimed to develop a user-friendly, online-based insight analysis tool for government, citizens, and SMEs in London. The objective was to improve procurement efficiency and competition. The experimenters wanted to combine their own existing data with the OC platform in order to cross-reference with new data, and to perform enhanced spending and tendering categorizations. In order to make government spendings and tendering processes visible and easily understandable, the experiment used the TinkerSpace tool,[6] the Asset Federation API,[7] and the OC's urban data observatory (UDO) [9]. TinkerSpace is one of the tools provided by the OC platform to simplify experiment development. This tool was used for the development of an interactive smartphone application where the users are provided with an intuitive overview of the categorized data. The Asset Federation API was used for adding data to the OC platform in order to enable third party developers to use such data. Furthermore, adding data to the platform made it easier to connect them to TinkerSpace, thereby reducing development complexity. Finally, UDO was used for validating that assets were actually pushed to the OC platform, and for presenting them to third-party developers.

In general, the experimenters were satisfied with the OC platform, and managed to use the services, although not always as expected. They even developed new functionalities for the TinkerSpace tool, which they made publicly available for anyone to use. Despite of the good impression, they experienced several inconveniences with the OC platform. They reported that documentation was lacking or needed to be significantly improved for external developers to properly grasp and utilize the OC platform. Understanding the services, the documentation and getting support from the OC technical team required considerable waiting periods which they deemed inefficient resource usage. In addition to documentation, they reported that access to historical data was paramount for their experiment, and it was inconvenient that the OC platform did not provide such a feature at that time. They managed to solve the issue by providing a custom persistence feature. On top of these issues, the experiment team suggested that authentication on the OC platform should be more consistent, since they had to confirm their log in every time they navigated between different portals of OC.

Final comments from this experiment team revolved around maturity level of the platform in general. They reported that the UDO needed to further develop customization, TinkerSpace needed to significantly improve the usability, and the OC platform APIs were being refactored during the experimentation period.

2) WearAQ: This experiment[8] utilized IoT, machine learning (ML), wearable technologies, and citizen participation for investigating human perception of air quality in cities. This was done through developing gesture recognition gloves, worn by pupils during a walk around the city. The children were taught simple hand gestures that would signify whether they felt the air was polluted or not. In parallel, the experiment team did simultaneous air quality measurements with high-quality pollution monitoring equipment. The collected data was combined with data from the OC platform, and London air quality data assets. By applying ML techniques on the generated dataset, the experimenters identified that there was a correlation between the children's perception of air quality, and the measurements of the pollution monitoring equipment.

In order to conduct the experiment, WearAQ utilized the Asset Federation API, Asset Discovery API, Scenario Tool, and the UDO. The Scenario Tool was initially used, to get inspiration on how to shape their experiment. The UDO was used for searching environmental and traffic data within confined geographical areas. The Asset Discovery API was used for extracting real-time data for their ML models. Finally, they wanted to use the Asset Federation API for inserting their refined datasets into the OC platform, so that future experimenters can work with their findings.

Even though the experiment was successful, the team reported several issues with the OC platform. In their own words, "there was a lot of documentation," but they reported that parts of it were missing. There were inconsistencies between different documentation pages, some cross-referencing links were broken, and it could be hard to decipher the documentation from a nontechnical perspective. They mentioned that it was difficult to grasp how to format parameters when using the RESTful APIs. They suggested that it might have been easier to comprehend if more usage examples were provided in the documentation. They also reported that error messages, returned from the APIs, were unclear. Apart from documentation issues, they reported that access to historical

[5][Online]. Available: https://organicity.eu/experiment/spend-network-2

[6][Online]. Available: https://docs.organicity.eu/tools/tinkerspace

[7][Online]. Available: https://docs.organicity.eu/api/Federation.html

[8][Online]. Available: https://organicity.eu/experiment/wear-aq-2

data was a lacking feature they had expected to be part of the OC platform, and they assumed that it was possible to manually upload assets directly through the Experimenter Portal. These missing features made the experiment more cumbersome, but they did mention that support from the OC technical team was good and swift, allowing the experiment to progress. As a general comment, they stated that the OC platform felt a little shaky due to multiple platform updates during the experimentation period (in order to provide new features and bug fixing).

3) Overall Experimenter Learnings: As mentioned previously, 25 experiments were conducted during OC's first open call. Through a grounded theory coding approach [21], we conducted two iterations of analyzing the evaluation questionnaires. First, we performed an initial coding by annotating relevant passages with summative descriptions. In the second iteration, we conducted focused coding thereby abstracting the descriptions into nine overarching categories [21]. We have provided the categories in Table I (in alphabetic order), and further discuss these in the remainder of this section. A detailed usability assessment is available online in [22].

From Table I, we see that the first deployment of the OC platform did not run smoothly, and this had consequences regarding how experimenters were able to use it in their experiments. Even though several issues and inconveniences occurred during the first experimentation period, the goal never was to have a fully fledged OC platform up and running from day one. Instead, we applied an experimentally driven approach, where we wanted to collaboratively develop the OC platform with the experimenters.

As a result, the first deployment of the OC platform should be perceived as a prototype of how an EaaS framework could be composed. This also means that the OC platform, as a whole, was not user-tested until it was made available to experimenters. The experimenters were aware of that the OC platform was not fully functional, and they were consequently prepared for experiencing issues. Knowing that experimenters would encounter problems, we set up several communication pipelines (e.g., email, Slack, or phone) so that experimenters could easily get in contact, and thereby progress their own developments. When issues and bugs were reported, we did our best to fix them immediately, but we had to prioritize which were most critical. As a result, some bugs were not corrected during the first experimentation period, and we suggested workarounds instead. We made sure to plan the implementation of the reported features so they would be implemented after the first experimentation period (i.e., historical data storage). According to Table I, it seems that the provided support pipelines have been satisfactory. From an internal OC perspective, support took up significantly more time than initially expected.

After the first open call ended, we significantly improved the OC platform by adding new functionalities. We have provided API access to historical data, and documentation has been significantly refactored with updated examples, step-by-step guides, and tutorials. Dead links, due to the continuous updates in the platform's documentation, have been fixed and coherency between different documentation locations has been improved.

In relation to documentation, experimenters reported this to be the culprit of most of their technical frustrations. Despite this, an interesting finding is that several experimenters reported that reading documentation and debugging took too much time, which they felt was both wasted and insufficient. We do not have specific figures on how much time experimenters actually spent on these activities, but empirical studies and our own experiences show that professional developers usually spend around 70% of their development time doing nothing but reading documentation and debugging [23]. The expressed frustration was therefore a result of the nontechnical nature of the experimenters participating in the open call. Since we do not have metrics for comparing time spent on actual programming, it is not possible to measure the efficiency of the OC platform's documentation, but according to the experimenters, this part took up a significant amount of their development time. In hindsight, this indicates that high quality documentation is of utter importance when exposing an EaaS platform to third-party developers.

B. Architectural Considerations for Platform Scaling

As we have commented in Section I, OC followed a centralized data architecture for both, storing and search. However, this approach may have scalability constraints, and hinder its adoption. In this sense, the ETSI[9] has created the context information management group, which is actively working on evolving the current definition of OMA-NGSI [24] and Orion Context Broker [25] implementation (referred to as Orion in the remainder of this paper), developed by the FIWARE EU Initiative.[10] The preliminary technical specification of the so called NGSI-LD has recently been published [26], and it proposes different options to deploy centralized and distributed architectures. In the following, we describe the different options that can be adopted and how they fit with the existing OC platform.

Since we have followed a centralized approach, the different OC services access the central Orion for any data-related action. As mentioned, this yields scalability limitations, and enforces to having a centralized architecture. In order to overcome such limitations, we are studying how to use NGSI-LD functionalities to enable hierarchical and distributed context information storage. Fig. 2 provides an overview of federation with distributed storage.

Similar to the current OC architecture presented in Fig. 1, the future approach will consider that different sites (i.e., cities) produce data that can be federated with a central OC instance. Cities will host local instances of Orion, that will be fed by context producers. Typically, the producers are services run by city service providers, that generate information which the city stores. Then, the site brokers notify the context registry about the locally stored data. Different to the current OC implementation, the registry does not store the data, but

[9][Online]. Available: http://www.etsi.org

[10][Online]. Available: https://www.fiware.org

TABLE I
EXPERIMENTER EXPERIENCES

Category	Summarized experiences	Improvements
Bugs	Reported bugs ranged from being inconsistencies in documentation (i.e., dead links) to not being able to create data assets through the Experimenter Portal, and experiencing that private assets where publicly available for third parties. Bugs could be small and insignificant to resulting in complete lock-down of developing an experiment. This means that bugs were present in all parts of the OC platform, and most experimenters experienced those at some point.	Cleaning up documentation, fixing bugs, altering existing functionality and defining new services. User and Experiment management were services heavily refactored in order to fix bugs and provide a higher degree of management granularity (see Section IV).
Data Availability	Data within the OC platform were outdated or not available when experimenters wanted to use them. The volume of data was too small, and it was challenging to find relevant data.	Improved data search through an Annotations service and a Reputation service (see Section IV).
Documentation	This category has been heavily commented throughout all evaluations, and seems to be the pivot point for most of the frustrations during the experiments. We have divided this category into seven sub-categories (ordered alphabetically): • *Ambiguity*: Ambiguous API naming conventions, abbreviations and contradictory documentation has been a hindrance. • *Error messages*: Vague, confusing, unhelpful and generic error messages were difficult to comprehend, and no documentation, on how to interpret the messages, was provided. • *Examples*: More and better usage and coding examples, as well as step-by-step tutorials are desired. • *Improvements*: Updating documentation more often, providing high-light examples, and use the Github[9] platform features even more would be of great value. • *Inconsistencies*: Parts of documentation was deprecated, had dead links, and errors like references to *localhost* was not corrected. Documentation could suddenly change and there was inconsistencies between different documentation. • *Readability*: Documentation was not easy to comprehend for a non-technical person. • *Unclear descriptions*: Documentation was not sufficiently elaborated and hard to follow, it was not clear how to format API parameters and it could be hard to understand how the OC services should be used and interact.	Recognizing that documentation was the main source of frustration, this part has been totally refurbished. The new documentation[10] first provides a general OC platform description for experimenters to become familiar with the modules and terminology. From such description, links are provided to different parts where technical information is provided. In addition, step-by-step tutorials have been elaborated covering the most relevant and recurring procedures. Finally, the APIs documentation has been reviewed to avoid inconsistencies.
Lack of functionality	Two features were lacking, which experimenters had expected to be available. This is API access to historical data, and the ability to provide filtering options when searching for data.	Historical REST-based data API introduced during first experimentation phase, and data filtering added to the UDO. WebSocket-based service developed for providing push notifications on asset updates (see Section IV).
Positive feedback	The OC platform was well designed and the available tools were both relevant and usable. Especially the graphical data searching (the UDO) was great for getting an overview of what data was accessible through the OC platform. One experiment team reported, that the authentication and authorization functionality enabled them to have a clear division between their own custom developed software and the OC services.	Annotations and Reputation services added to the UDO. Improved ability to interface directly with the OC platform APIs. Particularly evident in the User management and Annotations services.
Suggested new functionalities	Data visualization is quite simple in the OC platform, and is reduced to showing assets on a map in a browser. Experimenters would have liked more comprehensive data visualization features in order for them to play around with the data, and it could have simplified their custom developments. It was also reported that visualizations of historical data would have been beneficial. Finally, one experiment team reported that they wanted to have more fine grained authorization configurations in the APIs in order to control who could upload data to the OC platform.	The UDO was continuously expanded, and a historical data graph was added at the very end of the first experimentation phase. Fine-grained user and experiment configuration have been added through the User and Experiment management services.
Support	Level of support was satisfactory, and direct communication via phone, email or Slack[11] was fast and helpful. A few experimenters reported that Slack could be confusing due to multiple channels, and occasionally response times were slow.	Slack has been replaced by an open forum[12], based on the Q2A[13] platform, in order to reduce confusion and optimize support efforts.
Technical breakdowns	The Experimenter Portal produced error messages, changes could not be saved and invitations to participants would not be sent. The OC platform crashed regularly initially, functionality was often refactored, and experimenters had to adapt their existing codebase. When adding new assets to the platform, errors would arise when API attributes contained special characters.	Errors and issues were captured and major revisions carried out including: better functionality consistency, lower coupling between platform components, updating server infrastructure and architectural considerations.
Usability	From a usability perspective, experimenters were divided. Half of the experimenters found the OC platform easy to understand and use, while the other half found it cumbersome and unnecessary complex. The latter consisted primarily of experiment teams with a non-technical background. Regarding the UDO, most experimenters found it easy to use and helpful for understanding the concept of the OC platform. On the other hand, most experimenters had difficulties understanding and using the authentication and authorization functionality. Parts of the issues came down to understanding OAuth2.	A large part of the platform revisions have been pivoting around improving usability through step-by-step examples, expanding the graphical representation of User and Experiment management, reducing the steps needed to authorize against Keycloak, only expose relevant API functionalities etc.

only where the data can be found. Finally, a distribution broker acts as proxy to access the data, by checking the data location in the registry. As can be observed, in this approach no data duplicates are generated, thus solving scalability issues. In addition, it is possible to replicate the same architecture at different scales, as depicted in Fig. 2. This way, a site broker can internally act as a distribution broker. Furthermore, this approach will also enable cities to deploy local instances of OC with the required micro-services, while still being part of a larger deployment.

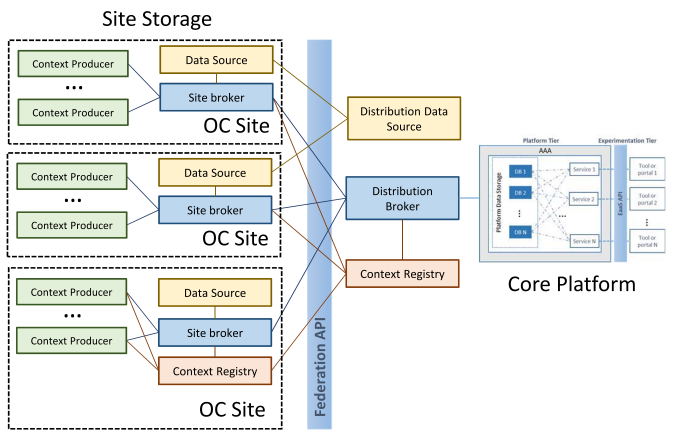


Fig. 2. Distributed federation architecture.

IV. Advanced EaaS Services

As depicted in Fig. 1, the OC platform design follows a three-tier approach, addressing data provisioning, platform management, and experimentation support. The Site Tier accounts for the data sources (e.g., cities) federated in the OC platform (named OC site). The Platform Tier holds the platform services, which are exposed for experimentation in the Experimentation Tier. The persistence within the Platform Tier relies on Orion, which is built around the NGSI 9/10 protocols. It manages entries, called assets, which can represent devices, places, buildings, and any other context information entities (including virtual objects). Each asset holds a set of typed attributes which can be updated on the fly [9]. The Platform Tier has been designed using loosely coupled and distributed components following the micro-service architecture pattern [27]. This allows tailored deployments involving only the mandatory, or baseline services, so that others (i.e., APIs, tools, and portals) can be optionally deployed.

The rest of this section covers improvements performed in the Platform and Experimentation Tiers. This tier makes use of the EaaS APIs (see Fig. 1), and it embraces tools and Web-based user interfaces (Web portals). Each EaaS service (i.e., API, tool, and portal) is implemented as a micro-service. In this sense, some of them are standalone applications, while others rely on baseline services, such as the Orion or the user management. For a more detailed description of the platform, the reader may refer to [9], which covers the EaaS services, which were used in the first open call. Based on feedback from experimenters in the first open call, we refactored and extended the OC services, and present these in the following sections.

A. User Management

Within OC, we cater to different stakeholder "roles," e.g., administrators, site managers, experimenters, and participants. An administrator is the technical administrator of the OC platform and, therefore, has access to information of all experiments at platform level. A site manager manages an OC site, which is a set of assets under a common administrative domain available for experimentation, e.g., the assets of a city. Each role is allowed to access different systems, which

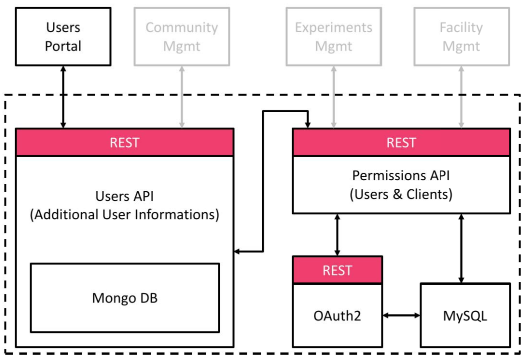


Fig. 3. User management APIs.

consists of an API and a portal. To handle this, we developed a user management system to handle authentication and authorization.

The user management API is based on OAuth2's API [28], with a single sign-on service, a *Users API*, and a *Permissions API* (see Fig. 3). They are used to manage accounts for users and clients (e.g., OC services). We use KeyCloak[11] as an OAuth2 server. To allow other OC services to manage accounts, we created the Permissions API, which acts as a configuration proxy for a more fine-grained access to the roles and permissions of accounts. Users must register at the server and verify their account by email. Afterward, a user can log into all OC services by using a single account.

The *Users Portal* allows users to manage their data, such as user and real names, nationality, email, age, etc. The Users API allows a quick and efficient user search for other OC services, since it uses MongoDB.[12] Some information like the username, email, and password is synced between the Users API and the KeyCloak server. Authentication and authorization are still handled by the OAuth2 API, the other OC services can use the Users API to manage accounts.

B. Experiment Management

One of the key aspects of the OC platform is the definition of an experimentation management system, able to fulfill the requirements of potentially heterogeneous experiments. This system has been implemented as a set of APIs exposed through three Web portals. First, the *Experimenter Portal* allows experimenters to manage all aspects related to experiment management and exploitation. Closely related, the *Communities Portal* enables the grouping of users, based on their interests and preferences, into the so called communities. Finally, the *Participant Portal* permits potential participants to manage the experiments in which they are enrolled. We briefly describe the functionalities of each of the Web portals in the following.

1) Participant Portal: This portal is used by users registered in OC, who have stated their willingness to participate in experiments, and it offers three main functionalities.

1) *Experiment Search*: All public experiments are visible to the participants, they can proactively select those taking

[11] [Online]. Available: https://www.keycloak.org
[12] [Online]. Available: https://www.mongodb.com

place close to them or that are considered interesting, according to the experiment information.

2) *Notifications*: Participants can be notified about any event happening in the experiments they are involved in, and they can be invited to new experiments. It is worth noting that all interactions happen in *an anonymous way*. Experimenters are not aware of the identity of participants.

3) *Enrolling Management*: Participants can decide to leave an experiment at any point. It is applied to both experiments, which they proactively joined, and those which they were invited in.

2) Communities Portal: Once users register in OC, they can complete their profile by providing more information about their interests and preferences. Through the Communities Portal, administrators and experimenters can filter users and create *communities*. A community is a virtual group linked to an experimenter or administrator, in a way that users and participants are not aware of the communities they belong to. Moreover, experimenters and administrators are not aware of the actual identity of the user in their communities, but they see simple indicators on the characteristics of their community's audience.

3) Experimenter Portal: This portal is the entry point of experimenters to the OC platform, and serves as a hub for the experimentation services. Following the micro-service design of the entire OC platform, the Experimenter Portal incorporates new services as independent views which may eventually be deactivated depending on the actual requirements. By using the Experimenter Portal, experimenters can perform the following tasks.

1) *Creation and Management of Experiments:* This functionality utilizes the user management API. Following the dominant concept in IoT experimentation, each experiment instance is seen as a virtual testbed, and uniquely identified throughout the whole platform. This way experiment datasets and management information can be isolated. In addition, at any moment experiments can be stopped/restarted and edited (e.g., name, description, etc.).

2) *Creation and Management of Assets*: Leveraging the public platform APIs, the Experimenter Portal provides a graphical UI to create and edit assets, in order for experimenters to become familiar with the data format. In particular, the graphical UI is a JSON editor able to check that the data follows the OrganiCity format, providing useful warning messages otherwise.

3) *Experiment Team Management:* Since co-creative experiments usually involve development teams rather than a single person, the Experimenter Portal also permits editing such teams. In this regard, at any moment the creator of the experiment can edit the team by adding or removing their members.

Apart from the aforementioned functionalities, the Experimenter Portal also provides views closely related to other services, and that can be used by the experimenters.

a) Annotations: The Experimenter Portal offers a UI to manage *tags* that can be applied on the assets created under each experiment. Experimenters can both select tags provided by the OC platform itself, or define new ones, available only for their own experiment. Tags are explained in more detail in Section IV-D.

b) Participants management and engagement monitoring: By using the communities created with the Communities Portal, experimenters can invite people to their experiments, and check whether or not invitations have been accepted. In addition, experimenters can define engagement metrics as a function taking into account different parameters. Then, during the execution of the experiment that function grows differently for the different participants according to the engagement level. For instance, a function can be defined as the amount of created data, number of readings, or the number of annotations added by a participant. Finally, using the notification functionalities provided by the platform, experiments can reward or incentivize participants.

C. Real-Time Notifications

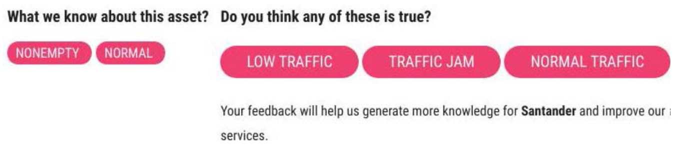
The requirement for real-time data is crucial for understanding and enhancing the smart city [29], which is why the OC project has chosen to emphasize the need for real-time data streams when performing experiments through the OC platform. This is both relevant for city administrators needing to optimize and predict city infrastructures but also for ordinary citizens who want to understand and affect the city [29]. Real-time access to city data can even become a collaboration interface between city administrators and regular citizens in developing the city. A specific example of such a collaboration is the application *Bus Santander*[13] that shows real-time estimates on when a bus arrives at a bus stop. The application was developed by ordinary Santander citizens leveraging open real-time data provided by the municipality. The application has become an augmented service for the public transportation, thereby making transport more seamless and essentially advancing the city through smart city technologies.

In the first instantiation of the OC platform, we did not provide a bi-directional, notification-based, real-time access to data but chose to provide access through a request and response approach leveraging the RESTful architecture of OC. This approach was sufficient for conducting experiments, but several experimenters requested the ability to have a bi-directional real-time connection to data. They wanted a way to get notifications when an asset changed instead of performing a continuous RESTful-based polling. Orion provides a subscription mechanism that is built as a WebHook.[14] An end-user needs to set up their own Web server, to which Orion can then send notifications. Even though this is a standard approach, many of our users were not programmers, and we therefore did not find this solution appropriate and easy to comprehend.

Additionally, some experimenters wanted to use state-of-the-art Web technologies in their experiments. As a result, we chose to investigate WebSockets since this technology fulfilled these requirements. FIWARE has even been testing WebSockets in relation to Orion, but development has ceased,

[13][Online]. Available: http://bussantander.truebaj.com

[14][Online]. Available: https://fiware-orion.readthedocs.io


What we know about this asset?
NONEMPTY
NORMAL
Do you think any of these is true?
LOW TRAFFIC
TRAFFIC JAM
NORMAL TRAFFIC

Your feedback will help us generate more knowledge for **Santander** and improve our services.

Fig. 4. Asset annotation interface in the UDO. Asset's annotation (left) and user feedback (right).

due to other parts of Orion being more critical.[15] Knowing that WebSockets was put on hold for Orion, we chose to develop our own, so that end-users can get notifications on asset updates. We developed a thin WebSocket layer on top of the WebHooks subscription feature. As a result, our WebSockets tool exposes the same features as Orion's subscription feature, meaning that we can only get notification on asset changes, and not on asset creation and deletion. Despite this shortcoming, the WebSockets tool still adds the ability to subscribe to asset updates in the OC platform.

D. Annotations Service

One of our main goals, after having provided experimenters with an initial set of urban data, was to increase the knowledge that can be assigned to such data by adding extra information on top. Information like that should be easy to spot, use, and add, as our users are not experts in fields like semantics or ontologies. Therefore, we based our design on the simple and well-established concept, used in social media over the past years, with which users can assign tags to assets and then search for information based on those. The Annotation API was built as a micro-service that maintains a tag-based taxonomy, as it is created by the users of the platform and allows for the addition, updating, and removal of tags to each asset of the OC platform.

We provide this functionality through OC's UDO, and users can answer simple questions on the presented data, by visually inspecting the data reported. Such an interface is presented in Fig. 4, where users provide feedback on a traffic monitoring sensor in Santander, based on data about vehicles passing by hourly. The responses of all users are displayed in the in left hand side of the figure.

To further increase the value of the information offered to users, we developed a set of services that operate on top of the generated annotations. Specifically, we developed a service that automatically adds annotations to the assets of an experiment using ML, based on provided training datasets. This service allows experimenters to setup and run cloud-based ML jobs to classify the streaming data of the platform in real-time. The process requires a set of initial training data for each of the classifications defined, and the result of the classification is stored as an annotation in the Annotation Service. For each job, the user needs to select a set of tags from the Annotation Service and submit the respective training data. The ML substrate of the system is implemented using JavaML [30], but is modular in design, i.e., other ML algorithms, e.g., Google TensorFlow [31], can be added in the future. In its implementation, we use the same principles of the WebSocket Tool and Orion's subscription mechanism to receive real-time updates on the data and trigger the analysis. It creates a subscription for a set of assets the user selects based on URIs and data types to receive updates. Additionally, this service provides a RESTful API so that users can submit external data for classification. If the asset submitted in both cases is available in OC, then the resulting classification is stored in the Annotation Service and propagated to the Orion. If the asset is not available in OC, the classification is simply returned as a response to the API request. More information on the mechanism used to generate the annotations is available in [32].

During the experimentation and the operation of the platform in the past months, users of OC have produced more than 42 000 annotations, using over 100 different tags provided either by OC or the experimenters involved.

E. Reputation

The *Reputation Service* uses direct and indirect information from the OC platform users, to make it easier to navigate inside the assets available on the OC platform. It calculates a *reputation* for each asset as a single rating that captures its popularity and usefulness to the end-users. Reputation can also be perceived as trust, by modeling the degree of trust end-users assign to each specific asset. We focus on the generic notion of reputation, that is public and combined, and not a personal/subjective one.

The Reputation Service is embedded in the UDO and uses statistical data from the Annotation Service. The reputation score is first calculated after a user has interacted with the assets for the first time. To generate a reputation score, a user has to either create an annotation for the specific asset, or to manually rate it. The rating information is submitted from the UDO to the Annotation Service and triggers an update to the reputation of the asset. When a user selects an asset, the UDO retrieves nonzero reputation statistics from Annotation Service and calculates the reputation score based on a weight-based model. This architecture is more efficient than a stand-alone reputation service, as it removes the requirement for an additional service, and allows for more dynamic values for reputation as they are calculated on request on the user's end. Furthermore, the overhead due to the update and retrieval of the statistics is negligible, as the UDO communicates internally with the Annotation Service with minimal network delays.

For modeling the reputation of assets, we employ a statistical-based model due to its simplicity, lightweight computational requirements and the extendibility by easily integrating new parameters when necessary. The reputation model is based on both subjective and objective parameters of the assets.

1) Opinion represented as a five-star rating.
2) *Usage of Statistics/Popularity:* How many times an Asset has been annotated by users? How many times has an Asset been rated by different users?
3) *Time of the Most Recent Action:* What was the last time that an Asset has been annotated?

[15][Online]. Available: https://github.com/telefonicaid/fiware-orion/issues/1181

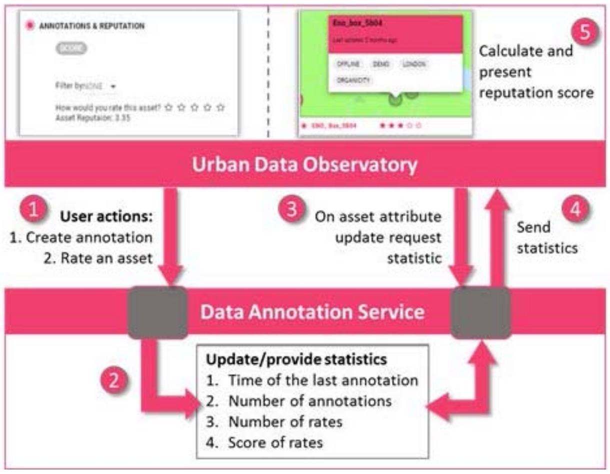


Fig. 5. Asset reputation interface in the UDO (user input and calculated value).

The final trust value is calculated as the average of all the parameters. The interface through which users can add their rating of an asset or view its reputation is presented in Fig. 5.

V. Conclusion

In this paper, we discussed the implementation of an EaaS framework within OrganiCity, and have presented our position regarding EaaS, i.e., to use existing IoT deployments in multiple cities in a federated manner to implement smart city prototype solutions. In this context, we presented our toolset, which enables EaaS in this application domain, in a scaled and integrated manner that has not been implemented previously. We have presented our design and implementation, discussed how OrganiCity and its toolset fit into the current smart city landscape, and presented several core components.

In order to validate this approach, we discussed learnings from deploying an EaaS framework in the wild, by running 25 experiments from independent experimenter teams. We also presented two specific experiments utilizing the OC platform as well as elaborated and discussed specific experimenter learnings, produced by using the first version of our platform. Our first findings indicate that this toolset has been utilized by the community and can be impactful. We discussed the ways in which the platform was used, where it fell short and how it has evolved through experimenter feedback. We additionally discussed results concerning the utilized architecture and scalability concerns, having used a more centralized architecture. From an architectural perspective, we can conclude that federation alternatives exist to handle potential scalability issues. In addition, following these approaches, any modification would be transparent to the systems implemented within the OC platform. We believe that this feedback, produced by such a scale of experimentation, will be very useful to the community currently working on similar aspects in the smart city domain.

OC is currently nearing its completion. The OC platform has been significantly refactored taking experimenter learnings into account, and the updated OC platform has been put to the test in a second open call following the same approach as in the first one. Results concerning this round of experiments are not included in this paper, but will be presented in our future work.

Acknowledgment

The authors would like to thank the experimenter teams and volunteers who participated in OrganiCity.

References

[1] S. Al-Nasrawi, A. El-Zaart, and C. Adams, “The anatomy of smartness of smart sustainable cities: An inclusive approach,” in *Proc. Int. Conf. Comput. Appl. (ICCA)*, Sep. 2017, pp. 348–353.
[2] O. Vermesan and P. Friess, *Building the Hyperconnected Society: Internet of Things Research and Innovation Value Chains, Ecosystems and Markets*, vol. 43. Aalborg, Denmark: River, 2015.
[3] M. Pallot and K. Pawar, “A holistic model of user experience for living lab experiential design,” in *Proc. 18th Int. Conf. Eng. Technol. Innov. (ICE)*, Jun. 2012, pp. 1–15.
[4] B. Pokrić, S. Krčo, M. Pokrić, P. Knežević, and D. Jovanović, “Engaging citizen communities in smart cities using IoT, serious gaming and fast markerless augmented reality,” in *Proc. Int. Conf. Recent Adv. Internet Things (RIoT)*, Apr. 2015, pp. 1–6.
[5] L. Phuluwa and M. Hattingh, “Understanding how the city of johannesburg metropolitan municipality’s social media platforms are perceived by young citizens,” in *Proc. IST Africa Week Conf.*, May 2017, pp. 1–10.
[6] I. Celino, G. R. Calegari, and A. Fiano, “Towards Talkin’Piazza: Engaging citizens through playful interaction with Urban objects,” in *Proc. Int. Smart Cities Conf. (ISC2)*, Sep. 2016, pp. 1–5.
[7] G. Marinic and W. Vanobberghen, “ECIM: European cloud marketplace for intelligent mobility,” in *Proc. Joint Doctoral Symp. Projects Showcase Held Part STAF Co Located Softw. Technol. Appl. Found. (STAF)*, Vienna, Austria, Jul. 2016, pp. 72–79.
[8] H. Schaffers *et al.*, “Integrating living labs with future Internet experimental platforms for co-creating services within smart cities,” in *Proc. 17th Int. Conf. Concurrent Enterprising*, Jun. 2011, pp. 1–11.
[9] V. Gutiérrez *et al.*, “Co-creating the cities of the future,” *Sensors*, vol. 16, no. 11, p. 1971, 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/11/1971
[10] H. Schuldt, *Multi-Tier Architecture*. Boston, MA, USA: Springer, 2009, pp. 1862–1865, doi: 10.1007/978-0-387-39940-9_652.
[11] L. Sanchez *et al.*, “SmartSantander: IoT experimentation over a smart city testbed,” *Comput. Netw.*, vol. 61, pp. 217–238, Mar. 2014.
[12] G. Coulson *et al.*, “Flexible experimentation in wireless sensor networks,” *Commun. ACM*, vol. 55, no. 1, pp. 82–90, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2063176.2063198
[13] T. Akiyama *et al.*, “FESTIVAL: Design and implementation of federated interoperable smart ICT services development and testing platform,” *J. Inf. Process.*, vol. 25, pp. 278–287, Mar. 2017.
[14] B. Cheng *et al.*, “FogFlow: Easy programming of IoT services over cloud and edges for smart cities,” *IEEE Internet Things J.*, vol. 5, no. 2, pp. 696–707, Apr. 2018.
[15] V. Gutiérrez, D. Amaxilatis, G. Mylonas, and L. Muñoz, “Empowering citizens towards the co-creation of sustainable cities,” *IEEE Internet Things J.*, vol. 5, no. 2, pp. 668–676, Apr. 2018.
[16] B. Guo, C. Chen, D. Zhang, Z. Yu, and A. Chin, “Mobile crowd sensing and computing: When participatory sensing meets participatory social media,” *IEEE Commun. Mag.*, vol. 54, no. 2, pp. 131–137, Feb. 2016.
[17] M. Talasila, R. Curtmola, and C. Borcea, “Crowdsensing in the wild with aliens and micropayments,” *IEEE Pervasive Comput.*, vol. 15, no. 1, pp. 68–77, Jan./Mar. 2016.
[18] L. G. Jaimes, I. J. Vergara-Laurens, and A. Raij, “A survey of incentive techniques for mobile crowd sensing,” *IEEE Internet Things J.*, vol. 2, no. 5, pp. 370–380, Oct. 2015.
[19] M. Balestrini, T. Diez, P. Marshall, A. Gluhak, and Y. Rogers, “IoT community technologies: Leaving users to their own devices or orchestration of engagement?” vol. 1, no. 1, 2015, Art. no. 150601.
[20] C. A. Le Dantec, M. Asad, A. Misra, and K. E. Watkins, “Planning with crowdsourced data: Rhetoric and representation in transportation planning,” in *Proc. 18th ACM Conf. Comput. Supported Cooperative Work Soc. Comput.*, Vancouver, BC, Canada, 2015, pp. 1717–1727.
[21] K. Charmaz, “Coding in grounded theory practice,” in *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*, 1st ed. London, U.K.: Sage, 2006, pp. 42–66.

[22] A. Heijnen *et al. D5.5 Usability Assessment of Organicity—First Open Call*. Accessed: Oct. 5, 2018. [Online]. Available: http://organicity.eu/resources-news/resources/
[23] R. Minelli, A. Mocci, and M. Lanza, "I know what you did last summer—An investigation of how developers spend their time," in *Proc. IEEE 23rd Int. Conf. Program Comprehension*, 2015, pp. 25–35.
[24] "NGSI context management," Open Mobile Alliance, San Diego, CA, USA, Rep., 2012. [Online]. Available: http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf
[25] FIWARE Foundation. (Jun. 2018). *Publish/Subscribe Context Broker—Orion Context Broker*. [Online]. Available: https://catalogue-server.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker
[26] *Context Information Management (CIM); Application Programming Interface (API), V1.1.1*, ETSI Standard GS CIM 004, 2018. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/CIM/001_099/004/01.01.01_60/gs_cim004v010101p.pdf
[27] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, vol. 1. Beijing, China: O'Reilly Media, Feb. 2015.
[28] D. Hardt, "The OAuth 2.0 authorization framework," Internet Eng. Task Force, Fremont, CA, USA, RFC 6749, Oct. 2012.
[29] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 112–121, Apr. 2014.
[30] T. Abeel, Y. V. D. Peer, and Y. Saeys, "Java-ML: A machine learning library," *J. Mach. Learn. Res.*, vol. 10, pp. 931–934, Apr. 2009.
[31] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16, 2016, pp. 265–283.
[32] A. Deligiannidou, D. Amaxilatis, G. Mylonas, and E. Theodoridis, "Knowledge co-creation in the OrganiCity: Data annotation with JAMAiCA," in *Proc. 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 717–722.

Dimitrios Amaxilatis received the M.Sc. degree in computer science from the University of Patras, Patras, Greece, in 2013, where he is currently pursuing the Ph.D. degree.

Since 2010, he has been with the Computer Technology Institute, Patras, participating in several international research projects. He was also a member of the founding teams of two technological startups (codebender.cc and Sensorflare). His current research interests include distributed algorithms, wireless sensor networks, home and building automation, smart city, and participatory sensing.

Dennis Boldt received the M.Sc. degree in computer science from the University of Lübeck, Lübeck, Germany, where he is currently pursuing the Ph.D. degree.

Since 2012, he has been a Research Assistant with the University of Lübeck. He has been involved in several national and international research projects, including SPITFIRE (FP7) and OrganiCity (H2020). His current research interests include peer-to-peer networks, full stack Web-based technologies, and network security.

Johnny Choque received the electronic engineering degree from the National University of Engineering, Lima, Peru, in 1995 and the Ph.D. degree (*cum laude*) in communications engineering from the University of Cantabria, Santander, Spain, in 2014.

Since 2000, he has been a Researcher with the University of Cantabria, researching in several Projects of Framework Programme and Horizon 2020 of EU. His current research interests include LPWAN, smart cities, Internet of Things, and blockchain.

Luis Diez received the M.Sc. and Ph.D. degrees from the University of Cantabria, Santander, Spain, in 2013 and 2018, respectively.

He has been involved in different international and industrial research projects. He is currently a Senior Researcher with the Network Planning and Mobile Communications Laboratory, University of Cantabria. His current research interests include resource management in wireless heterogeneous networks and Internet of Things service provisioning.

Etienne Gandrille received the Ph.D. degree from the University of Grenoble, Grenoble, France.

He is a Researcher with the Commissariat à l'Énergie atomique et aux Énergies alternatives, Paris, France. His current research interests include Internet of Things, smart cities, and open data. He has been involved in several international research and innovation projects, including SocIoTal (FP7), OrganiCity (H2020), BigClouT (H2020), and Brain-IoT (H2020).

Sokratis Kartakis received the B.Sc. (Hons.) and M.Sc. degrees in computer science from the University of Crete, Heraklion, Greece, in 2006 and 2008, respectively, and the Ph.D. degree in computer science from the Department of Computing, Imperial College London, London, U.K., in 2017.

He has been involved in several international research projects collaborating with Yale University, New Haven, CT, USA, NEC, Tokyo, Japan, and ICS-Forth, Crete, Greece. Since 2017, he has been an IoT Research Scientist with Intel Labs Europe, London. His current research interests include edge signal processing and machine learning, and communication optimization for Internet of Things.

Georgios Mylonas received the Ph.D. degree from the University of Patras, Patras, Greece.

He is a Researcher with Computer Technology Institute and Press "Diophantus," Rio, Greece. His current research interests include wireless sensor networks and distributed systems, Internet of Things, and pervasive games. He has been involved in the AEOLUS, WISEBED, Smartsantander, AUDIS, and OrganiCity projects. He is the Coordinator of the Green Awareness in Action H2020 project.

Lasse Steenbock Vestergaard received the M.Sc. degree in information science from the University of Aarhus, Aarhus, Denmark. He is currently pursuing the Ph.D. degree at Alexandra Institute, Aarhus, Denmark, focusing on IoT prototyping within smart cities.

He has been involved in national and international research and innovation projects, including OUTSMART (FP7), SmartSantander (FP7), CityPulse (FP7), OrganiCity (H2020), and Synchronicity (H2020). His current research interests include creative coding, API usability, and Internet of Things prototyping.