IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 28, 2020, accepted December 17, 2020, date of publication December 21, 2020,
date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3046286

Fostering IoT Service Replicability in Interoperable Urban Ecosystems

**LUIS DIEZ, JOHNNY CHOQUE, LUIS SÁNCHEZ,
AND LUIS MUÑOZ, (Senior Member, IEEE)**
Communications Engineering Department, University of Cantabria, 39005 Santander, Spain

Corresponding author: Luis Diez (ldiez@tlmat.unican.es)

This work was supported in part by the European Union's Horizon 2020 Programme [SynchroniCity (Delivering an IoT enabled Digital Single Market for Europe and Beyond)] under Grant 732240, and in part by the Spanish Government (Ministerio de Economía y Competitividad, Fondo Europeo de Desarrollo Regional, MINECO-FEDER) through the project FIERCE: Future Internet Enabled Resilient smart CitiEs under Grant RTI2018-093475-AI00.

ABSTRACT Worldwide cities are involved in a digital transformation phase specially focused on sustainability and improving citizen's quality of life. However, such objectives are hard to achieve if the migration of the urban processes are not performed following a common approach. Under the paradigm of smart city, different Information and Communication Technologies (ICT) have been deployed over urban environments to enable such digital transformation. However, actual implementations differ from one city to another, and even between services within the same city. As a consequence, the deployment of urban services is hindered, since they need to be tailored to each city. In addition, the isolation of urban services obstructs its optimization, since it cannot harness contextual information coming from other services. All in all, it is necessary to implement tools and mechanisms that allow us to ensure that city solutions and their vertical services are interoperable. In order to tackle this issue, different initiatives have proposed architectures that homogenize the interaction with smart cities from different angles. However, so far the compliance with such architectures has not been assessed. Having this in mind, in this work we present a validation framework, developed under the umbrella of the SynchroniCity project, which aims to verify that interfaces and data exposed by cities are aligned with the adopted standards and data models. In this regard, the validation framework presented here is the technical enabler for the creation of an interoperability certificate for smart cities. To assess the benefits of the validation framework, we have used it to check the interoperability of 21 smart city deployments worldwide that adhered the SynchroniCity guidelines. Afterwards, during an open call a total number of 37 services have been deployed over such SynchroniCity instances, thus confirming the goodness of uniform and validated smart cities to foster service replicability.

INDEX TERMS Internet of Things, smart city, validation, replicability, interoperability, deployability.

I. INTRODUCTION

The pervasive presence of Information and Communication Technologies (ICT) in all realms of society is allowing the evolution of legacy services and the definition of new ones. In the particular case of urban environments, the maturity level reached by Internet of Things (IoT) technologies is leading to its adoption in most of the social-economic and industrial activities. In particular, IoT permits continuous systems monitoring to optimize processes management. In addition, considering the complexity of urban ecosystems, and how the different services interplay, it is necessary to foster interaction between verticals to ensure a global improvement of the citizen's quality of life.

However, most of the urban services that leverage IoT technologies are deployed as isolated systems, hindering the interoperability between them (i.e. energy, traffic or urban transportation). On top of that, the adoption of proprietary solutions suppose an additional technological barrier to enable service interoperability. Furthermore, as cities provide different interfaces, service providers cannot simply replicate their solutions, but it is necessary to adapt them to the different urban scenarios. In this context, it becomes desirable defining a set of well-defined interoperability mechanisms

The associate editor coordinating the review of this manuscript and approving it for publication was Arif Ur Rahman.

to overcome such limitations. In addition, the data generated by urban services is becoming more and more demanded. While so far cities have followed an open-data approach, this data sharing paradigm is likely to change in the short term. Data generated by one service may be highly valuable for other service providers, due to the potential interaction. This justifies the definition and consolidation of an IoT data market to permit urban stakeholders to control the sharing policy they want to apply over their datasets.

In order to define how stakeholders interact with smart city infrastructure, in the following we differentiate between *urban-data providers* and *urban-service providers*. The former group would inject data into the smart city platform exploiting a southbound interface, and the latter consumes data through the smart city northbound interfaces to provide a service to final users (e.g. citizens, municipality, etc.). It is worth noting that a given stakeholder can play both roles, urban-data and urban-service provider.

The new data-management paradigm requires that the interaction between the different stakeholders, as well as between them and the smart city infrastructure, is well defined. On the one hand, urban-data providers require that the smart city interfaces follow well known specifications, so that data coming from IoT devices can be easily adapted without needing to develop tailored solutions for every city. On the other hand, urban-service providers (i.e. urban-data consumers) need that data can be acquired by their service, also without requiring costly customized adaptations.

In order to answer those needs, the European *SynchroniCity*[1] project has proposed an architecture that aims to promote the adoption of a common technical foundation for smart cities, which enables service interoperability and replicability. To this end, *SynchroniCity* took an inclusive approach putting together cities and private stakeholders in the architecture definition. This way the proposed framework seeks providing a solution (1) easy to adopt by cities, even when they already have existing solutions, and (2) foreseeable in the way urban-data and urban-service providers can interact with it, so that it simplifies interoperability and service replicability.

Although the architecture definition is a necessary condition, actual interoperability also requires the validation, or certification, of the interfaces exposed by the smart cities to ensure a smooth interaction with the different stakeholders. To that end, based on the *SynchroniCity* adopted standards, we have defined, developed and deployed a validation framework that verifies the correctness of smart city interfaces. This validation covers access permission, functionalities exposed by Application Program Interfaces (APIs), as well as data models.

The *SynchroniCity* adoption has followed a 2 phases approach. First, it was deployed in 8 cities (Antwerp, Carouge, Eindhoven, Helsinki, Manchester, Milan, Porto, and Santander) involved in the architecture definition, hereinafter Reference Zones (RZs). Afterwards it has been extended by means of an open call to involve both other cities, and urban-data and urban-service providers. During the open call 16 providers have deployed their solutions on top of the proposed architecture both in the RZs and in other cities where the framework has been replicated. In both phases the validation framework has been applied to the different *SynchroniCity* instances, both RZs or external cities, to foster a correct adoption of the framework, and ultimately to ensure the efficient and effective deployment of urban solutions.

Altogether, this work aim to technically describe the interoperability validation framework. In this sense, we consider both the standards and data models that the smart city framework adopts, and how it is configured, to perform systematic validation over a set of cities. In order to validate the added value brought by the framework, we also describe and discuss the experiences of applying it to real-life smart city deployments during the open call. The discussion includes both quantitative results based on technical integration of services and data generation shaped by the adopted models, and qualitative analysis based on the feedback of companies involved in the open call. In the following, we enumerate the contributions of this paper:

- Overall description of the *SynchroniCity* architecture to identify interoperability mechanisms to be validated. It is worth noting that the *SynchroniCity* architecture is not a contribution itself, but its description is necessary to enumerate which interfaces are going to be validated.
- Description of the interoperability validation framework, including the overall systematic validation approach and its technical implementation.
- Assessment of the validation framework within an open call involving several cities and companies.
- Use of the validation framework for systematic endorsement of new services, assessing their relationship with the *SynchroniCity* architecture, and interaction with standards. This validated the service replicability on top of the *SynchroniCity* architecture, giving proofs of the acceptance of the *SynchroniCity* architecture.

The rest of the paper is structured as follows. Section II presents an overview of the related works, and the relationship with the one presented in this paper. Then, in Section III we briefly describe the *SynchroniCity* reference architecture, which has served to define and implement the validation framework. Afterwards, the validation framework is introduced in Section IV, describing the overall validation flow and technical aspects. Then, Section V presents the validation results obtained from using the framework over a set of smart city deployments in different countries. Finally, we conclude the paper in Section VI summarizing the main outcomes, and providing an outlook or our future work.

II. RELATED WORK

Along this section we will revise the literature related to IoT interoperability in general. As we will see, the concept of

[1] https://synchronicity-iot.eu/

interoperability has different approaches, embracing architecture, devices, or data. We will cover all of them to assess the novelty of this work, and identify how the tool set described herewith complement existing works. In spite of the large amount of literature related to smart cities, we focus on those pieces of work related to smart city architectures and interoperability. In this regard, we do not focus on data collection, the reader may refer to [1] for an in depth analysis sensing possibilities for smart cities. Similarly, the literature review does not tackle urban services, which would be deployed on top of the smart city architectures. In this sense, automation techniques that have gained relevance, such as those based in artificial intelligence and machine learning [2]–[4], are not studied.

In the last years, interoperability in the context of smart cities has been tackled from different angles. In the literature, one can find analytical papers that address studying the smart city structure and interoperability objectives. In this sense, in [5] smart city infrastructures is analyzed to identify common entities, while Antonić *et al.* [6] analyze traffic load of different urban verticals. Following this line, in [7] and [8] interoperability definition and requirements specification are addressed for the IoT ecosystem in general, and smart city in particular, respectively. In addition, other works have even tackled interoperability of smart cities with other systems. One example can be found in [9], where Pradhan investigates interoperability between different ICT and smart cities to carry out Humanitarian Assistance and Disaster Recovery (HADR) operations.

Having in mind the particularities of smart cites, different approaches have been proposed to enable such interoperability. Some works suggest adopting common designs [10], while others opt for adaptation layers to interconnect cities and services. For instance, in [11] Lopes *et al.* propose a System of Systems (SoS) middleware to interconnect services, and the idea of the hub-centric approach to aggregate various cities' data is suggested in [12]. Worth of attention are also the works proposing simulation tools to develop city services [13], where the idea of interoperability is assumed in the simulation models.

In parallel to the academia, standardization bodies have addresses the interoperability issue in smart cities. The ITU-T Focus Group on Smart Sustainable Cities (FG-SSC) [14] proposes a basic reference architecture for smart cities embracing application, data and support, network, and sensing layers. Similarly, the ISO/IEC JTC1 study group on Smart Cities [15] provides a layered smart city architecture also considering business aspects. In particular the following layers are defined: business, data, cloud and network, sensing, and security layers. Another layered architecture is proposed by the Alliance for the Internet of Things Innovation (AIOTI) [16], embracing application, network, and IoT layers. AIOTI seeks standardization, interoperability and policy issues for economic development, and growth of digital markets. As for data access and management, the road-map is being defined by the ITU-T Focus Group on Data Processing and Management to support IoT and Smart Cities and Communities (FG-DPM) [17]. Finally, the U4SSC initiative [18], coordinated by ITU and UNECE, advocates for public policies that allow using ICT to facilitate transition of smart, and sustainable cities. As we will see in the next section, the *SynchroniCity* platform follows a similar layered approach, so that it may be mapped to the ones defined by the aforementioned reference architectures.

Meanwhile the projects and initiatives enumerated above focus on the architecture design, there have also been projects that provided actual implementations. Although all of them present differences, interoperability is a recurrent issue. For instance, the *symbIoTe* project [19] seeks the definition of an abstraction layer to access IoT information from potential heterogeneous and isolated IoT islands. In a similar way, the *Fed4Fire* [20] and *Fiesta* [21] projects present some similarities with *symbIoTe*, since they addresses a uniform access to the underlying IoT infrastructures, but they are rather focused on Experimentation-as-a-Service (EaaS). In this realm, *OrganiCity* project [22], [23] is probably the most featured one, since it provides a wide range of tools to foster service co-creation, embracing all the stakeholder involved in urban services. *SoCiotal* project [24], similar to *OrganiCity*, puts the focus on citizens, developing an environment that allows them to control their IoT devices. Also under the EaaS umbrella, the *Festival* project [25] aims to develop federated environments to foster IoT experimentation.

As can be observed, there are significant differences between the aforementioned initiatives and the one presented for which the validation framework is developed. From an architectural perspective, most of the mentioned projects focus on interaction with IoT devices (i.e. lower-bound) and/or experimentation. Opposed to that, *SynchroniCity* is intended to provide both lower- and upper-bounds to be used by data and service providers, and thus the validation framework tackles both interfaces. Most importantly, none of the mentioned projects addresses interoperability assessment.

Nevertheless, there exist projects that also emphasize interoperability. For instance, the *ESPRESSO* project [26] promotes common metadata structures and open standards to avoid vendor lock-in. Similarly, the European Innovation Partnership on Smart Cities and Communities (EIP-SCC) [27] brings together different stakeholders, and proposes a reference architecture inspired in *ESPRESSO*. Similarly, *BIG-IOT* project [28] aims integration and interoperability, although the proposed solution uses its own interfaces, instead of open standards. In addition, the proposed architecture is not agnostic of the underlying IoT infrastructure, so hindering service replicability. It is worth mentioning *WiSe-IoT* [29], a Europe-Korea collaboration which focuses on interoperability and federation of IoT systems. Although these frameworks pursue some of the *SynchroniCity* objectives concerning interoperability, they did not develop means to ensure such interoperability.

Table 1 summarizes the main characteristics related to urban service replicability of the projects and initiatives

TABLE 1. Summary of projects and standardization initiatives that propose smart city architecture.

Initiative	North/South bounds	Implem.	Interop.	Validation
[14]	✓/✓	✗	✓	✗
[15]	✓/✗	✗	✗	✗
[19]	✗/✓	✓	✓	✗
[20]	✓/✓	✓	✗	✗
[21]	✓/✓	✓	✗	✗
[22]	✓/✓	✓	✗	✗
[24]	✓/✓	✓	✗	✗
[25]	✓/✗	✓	✓	✗
[26]	✓/✗	✓	✗	✗
[28]	✓/✓	✓	✓	✗
[29]	✓/✓	✓	✓	✗

that propose a smart-city architecture. The columns of Table 1 indicate: whether the architecture defines north- and southbound interfaces; if it has been actually implemented; whether it pursues smart city interoperability to foster urban service deployment; and whether interoperability validation has been implemented. As can be observed, although many initiatives foster interoperability the validation of the mechanisms to enable it has not been tackled before.

If we specifically focus on urban data, we can find works devoted to interoperability. Initially, most of them focused on common access to data from IoT devices. In this regard, Sharma *et al.* [30] presented a catalog of middleware and a framework to develop data schemas for IoT devices. Similarly, in [31] the authors propose a solution for the storage and retrieval of heterogeneous IoT sensor data in a uniform way, while the service *InterSensor* is described in [32] to provide web access to sensor data in an homogeneous way. It is also worth mentioning the framework *rsiHub* [33] which aims to easily provision IoT services that are interoperable to certain requirements. Elaborating the idea of IoT interoperability, in [34] Kim *et al.* proposes IoT testing as a service (IoT-TaaS) to ensure such interoperability. As can be observed, these works differ from ours as they specifically focus on interoperability of IoT devices, rather than smart city deployments.

Another group of works pay special attention to the semantic interoperability, including IoT data. In this regard, the authors in [35] enumerate the issues and requirements to develop semantic interoperability, while Gyrard *et al.* [36] present a survey of ontology-based software tools for semantic interoperability. More specifically, semantic interoperability applied to IoT is studied in [37] and [38] through objects profiling and annotations. On top of it, some works have proposed methodologies to develop semantic interoperability for IoT, such as *SEG3.0* [39].

A more practical approach to enforce this kind of interoperability can be found in [40], where the authors define interworking proxies to accomplish semantic interoperability between architectures based on Next Generation Service Interface (NGSI) and *oneM2M* standards. In a similar way, the *bIoTope* ecosystem [41] aims to implement interoperability between systems, provided they expose certain APIs and data models.

Although the works tackling data interoperability share some of our goals related to well-defined data and information models, they have a different scope. The mentioned works focus on analyzing data commonalities, so that different systems can interact. On the other hand, we aim to validate that an actual system (i.e. smart city) exposes certain standards, covering both APIs and data. Having this in mind, both approaches may be complementary, so that the tools presented in this work would be used before proceeding with the systems interoperability.

Worthy of attention are also the efforts conducted within the World Wide Web Consortium (W3C) related to the so called *web of things*. In [42] Blackstock and Lea defined the interoperability levels. Although this approach was validated through its adoption by different solutions, there is not a tool or framework that validates such interoperability. This definition has been further extended in [43] (Section 2.3) for IoT data semantics.

Finally, we have found in the literature a set of works providing tools to enable interoperability. Hernández *et al.* [44] describes a framework to specify adopted standards. Similarly, a model based, lightweight middleware to simplify interoperability of IoT services is described in [45]. Although the scope of these works are different from ours, the formal definition of adopted standards could be used by our framework in the future.

Additionally, some works propose implementations to interconnect systems, such as the IoT broker based on Message Queuing Telemetry Transport (MQTT) in [46], or *Hypercat* [47], which revolves around the concept if IoT hub. It is also worth mentioning the development presented in [48] by Zarko *et al.*, where a set of mediation services have been developed to access resources in a uniform way in the context of IoT, so embracing both semantic and syntactic interoperability. Other works, address specific use cases, such as integration of smart cities and Federated Mission Network (FMN) within NATO context [49]. As can be seen, all these works assume that the exposed APIs and/or used data models follow a known specification. On the contrary, the tool set presented in this work aim to validated that such assumption is correct.

All in all, the tools described in this paper suppose a step beyond the state-of-the-art which can be integrated with other solutions related to interoperability.

III. SynchroniCity ARCHITECTURE

This section introduces the logical components of the *SynchroniCity* reference architecture. The main purpose of the architecture is the definition of a set of Minimum Interoperability Mechanisms (MIMs) to ensure that interaction with it is predictable, and city services can be easily replicated. With the MIMs *SynchroniCity* aims to find a trade-off between service interoperability, and requirements imposed to the architecture adopters. On the one hand, the interoperability simplifies urban-services replication over different instances. On the other hand, the MIMs seek keeping the entry-barrier low to ease the adoption of the architecture by stakeholders and cities with existing technological solutions.

The architecture definition leverages commonalities of the ones presented in the previous section. Differently to former solutions, the *SynchroniCity* definition is based on actual cities' requirements and their existing technical implementations. Concretely, the architecture has been co-defined with 8 cities around Europe, so-called RZs, which have smart city solutions, or are in the process to implement them. In turn, during the process definition each RZ was represented by a member of the municipality as well as a technical team, thus ensuring that the technical solution is fully aligned with city needs. After the deployment in the RZs, the open call pilots replicated the same architecture in a number of cities to deploy their services. Detailed information of open call pilots and architecture replication is provided in Section V.

As a first step, a use-case analysis was performed in each RZs to identify a set of general requirements. This analysis permitted us to know the urban services each RZ was interested in, and to point out technical aspect common to all RZs. The reader may refer to [50] for a detailed description of the requirements analysis. This first step gave rise to a number of generic MIM, which are summarized as follows:

- *Southbound interfaces*: it covers the APIs that allow the interaction of *SynchroniCity* with the IoT devices and data sources of the cities.
- *Northbound interfaces*: it includes the APIs that enable urban services to access data in different ways.
- *Data management*: it embraces data storage and context management, and it also includes data models, format and semantics.
- *Security and privacy*: this module comprises well-known Authentication, Authorization and Accounting (AAA) functionalities.
- *IoT data marketplace*: it makes use of the aforementioned modules to provide the required functionalities that enable business-related interactions.

Afterwards, those RZs where smart city solutions existed were analyzed to identify similarities. Thus, the architecture definition tried to cover most of these commonalities, while keeping a generic approach to allow each RZ to implement the required architectural blocks reusing their current solutions [51].

This process permitted us to define a number of logical components that implement the MIMs, which are described in the following sections. As can be observed in Fig. 1, from an architectural perspective the APIs are grouped following the modules described before. First, the southbound interfaces include functionalities to manage context data (i.e. create, update, delete). Then northbound modules embraces those APIs that permit access to context and historical data. Around all the interactions with the architecture, the authorization module ensures that security and privacy policies are applied. Finally, the interfaces exposed by the northbound are exploited by data consumers, whose access are managed by an IoT data marketplace. It is worth pointing out that the reference implementation proposed by *SynchroniCity* [51] is deeply rooted in the FIWARE generic enablers.

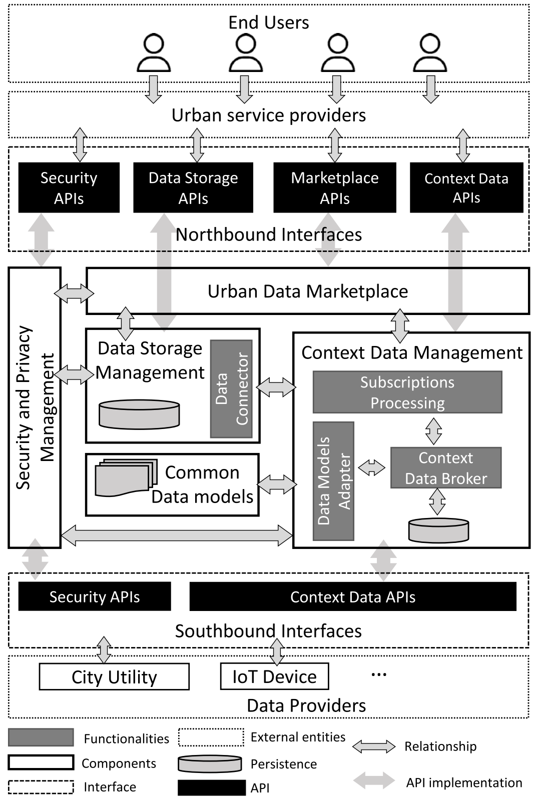

FIGURE 1. Simplified representation of the SynchroniCity logical architecture.

A. CONTEXT DATA MANAGEMENT

It manages the context information coming from IoT devices and other public and private data sources, providing a context data access through a uniform interface. The Context Data Management (CDM) abstracts any information related to real world into "context entities", which are made of attributes and metadata. This component provides functionalities to enable access to different data sources and analyze context information. In this regard, NGSI has been adopted as standard to provide data context information. Although, OMA-NGSI [52] was taken as initial definition, the ETSI-CIM working group [53] has elaborated a new standard usually referred to as NGSI Linking Data (NGSI-LD), that includes linking data. This new standard is the natural evolution to be adopted by the architecture.

Specifically, the CDM exposes the following interfaces:

- *Manage context*: it provides methods for synchronous creation, modification and deletion of context entities, which can be issued based on their attributes. In addition,

it is possible to either update or remove entire entities, or only selected attributes from them.
- *Query context*: this API provides methods to discover and retrieve entities by using different filters. It is also possible to perform progressive search by also applying filters over attributes' metadata. A query consists of a list of statements, each expressing a matching condition.
- *Subscription*: it provides methods to define and manage subscriptions to asynchronous notifications upon entities updates. The API permits defining subscription conditions based on attributes values, as well as thresholds or ranges. It is also possible to define the attributes that the subscriber wants to get notified about. This way, it allows subscription to specific information pieces, so that when matching entities are updated subscribers receive an asynchronous notification with the required information.

It is worth pointing out that the CDM is used by both south- and northbound interfaces. In this sense, this component can be exploited for both creation and retrieval of information, by urban-data and urban-service providers respectively.

As can be seen in Fig. 1, the CDM consists of various modules that implement the aforementioned functionalities. In addition, the CDM subscription functionalities permit generating historical records in the Data Storage Management (DSM) component described below.

B. DATA STORAGE MANAGEMENT

The DSM provides functionalities related to storage of historical data series and the means to access them. In order to anticipate the adoption of the ETSI-CIM standard, an specific API was defined for data access that, while still relying on OMA-NGSI standard, provided the same functionalities described in [53] for data access.

First, the DSM implements a connector that interacts with the CDM to capture the evolution of attributes of the context entities (using notifications), and store it in an appropriate way. Then, the DSM exposes an API implemented that offers methods to retrieve both historical data in general, and time series in particular. In addition, the API allows filtering historical data of attributes using both paging parameters and temporal limits. Detailed information is provided in the validation descriptions presented in Section IV.

C. COMMON DATA MODELS

While NGSI provides a defined data structure, the data models establish the data ontology and meaning of each field. Starting from broadly adopted data models, *SynchroniCity* has adopted a catalogue of data models according to actual city and service providers' requirements.

As commented before, data model plays an important in the smart city interoperability. By having a common data model definition, services can consume pieces of information in a uniform and predictable way, thus easing service replicability and deployability. Instead of following a clean-slate approach, *SynchroniCity* has leveraged broadly accepted data models, adapting and extending them according to actual city-services' needs. In particular, the following initiatives have been analyzed:

- The IoT Big Data Harmonised Data Models developed by GSMA [54].
- The FIWARE Data Models developed by the FIWARE community.[2]
- The *Schema.org* data models[3] developed by the browser and search engine communities.
- Ontologies promoted by the European standards community, particularly SAREF.[4]

The first two ones are closely related, existing discrepancies on the adopted data models and terminology. From them, FIWARE data models were selected as baseline for the *SynchroniCity* ones, due to the following reasons:

- They are based on the OMA NGSI information model and ready to be consumed using the NGSI API.
- They leverage data models already published by GSMA and reuse concepts coming from *schema.org* and SAREF, thus enabling interoperability and reusability.
- At least 5 of the RZs were already exposing data using NGSI APIs and/or the FIWARE data models.
- This initiative is endorsed by a growing community, which somehow ensures that the *SynchroniCity* extensions and definitions can be sustained in the long term.
- Also related to the future, FIWARE data models are being already adapted to the NGSI-LD ETSI standard, so that its adoption simplifies the overall transition to that standard.

Taking the FIWARE data models as a starting point, *SynchroniCity* has defined a procedure to further extend the adopted models and to define new ones. It is worth noting that, during the whole process, the *SynchroniCity* technical team has tightly interacted with the FIWARE community. This way, extensions and modifications of the adopted models have been also submitted to FIWARE repositories and, in most of the cases, accepted.

Although FIWARE data models are taken as baseline, not all of them have been adopted from the beginning. Contrarily, the adoption has followed an iterative process based on actual needs expressed by cities and providers. This way, when the existing data models did not fit their needs, RZs technical teams or open call pilots indicated the issue. Then, when modifications were not possible without breaking back-compatibility or the current adopted data models did not considered that type of data, FIWARE repository was revisited to find alternatives. Eventually, when necessary, new data models were defined and approved following a consensus approach.

Table 2 summarizes the adopted data models, according to the vertical service or category they belong to. As can be

[2] https://www.fiware.org/developers/data-models/

[3] https://schema.org/

[4] https://ontology.tno.nl/saref/index.html

TABLE 2. Summary of SynchroniCity data models†.

Vertical/Category	Models	Adopted	Defined
Environment	2	2	0
Point of Interest	4	2	2
Transportation	10	5	5
Weather	2	2	0
Mobility	17	0	17
Parking	4	4	0
Parks & Gardens	2	2	0
Waste	3	2	1
Device	2	2	0
Queue Monitor	1	0	1
Energy	3	0	3

†https://gitlab.com/synchronicity-iot/synchronicity-data-models

observed, most of the brand-new data models correspond to urban mobility, while a few ones are within transportation and energy categories. As for the urban mobility case, the reason was that some cities and open call pilots required General Transit Feed Specification (GTFS) based data models, which had not been considered in the existing definitions.

D. SECURITY AND PRIVACY MANAGEMENT

Security and Privacy covers all the security aspects related to data, IoT infrastructure and platform services, which underpin the applications and services of the cities. In this regard the architecture relies on the broadly adopted OAuth 2.0 standard [55]. The more relevant APIs exposed by this module are described below:

- *Identity and authentication API*: it permits creating, importing, retrieving, updating and deleting users. Additionally, it implements the authentication methods. As can be seen, this API is not intended for external users, but it provides smart city management functionalities.
- *Policy management*: this API allows managing policies, including the creation, update, retrieval and deletion of such access policies. Each policy defines a set of rules and subjects for those rules, which are afterwards applied to users. Again, this API is not used by platform users directly, but by platform managers.
- *Authorization and accounting*: it exposes methods to grant/deny permission to access resources, and to perform access request. In particular, this API implements the OAuth 2.0 authentication flows to allow users interacting with the urban platform.

E. DATA MARKETPLACE

Marketplace supports business interactions between data suppliers that are part of the *SynchroniCity* ecosystem and consumers. Its main aim is to foster data transactions to make reality its commercial viability in a sustainable way by developing a considerable added value whose scope goes beyond traditional rights-based licensing models of data sets. It contains a set of modules aimed to overcome the traditional data sharing barriers, boosting the confidence of data consumers and providers. In more detail, the data marketplace implements a hub to enable digital data exchange of urban data and IoT capabilities, providing features in order to manage asset catalogues, orders, revenue, billing and charging management, etc. These functions will support the creation of innovative business models. As reference implementation, *SynchroniCity* adopts FIWARE/TM Forum Business API Ecosystem[5] and evolves it.

IV. ASSESSING INTEROPERABILITY: METHODOLOGY AND VALIDATION TOOLS

In this section we describe the validation framework developed to assess the interoperability mechanisms defined by *SynchroniCity*. This framework is made of a set of tools, which permit us to systematically validate multiple *SynchroniCity* instances, as a first step to provide an interoperability certificate. It is worth noting that the developed software has been publicly available in a public Git repository.[6]

The overall validation flow is depicted in Fig. 2. First, the framework is configured with information related to each of the instance to be validated. Additionally, it is possible to define specific configuration for each service in an instance, when multi-tenancy deployment is used. The configuration information includes, for each pair city/service, the Uniform Resource Locators (URLs) of the exposed APIs, as well as user credentials and available authentication flow.

Then, when the validation is initiated a set of routines are executed over each instance. These routines are defined as a collection of queries applied over the city' APIs to cover their functionalities. In all cases, the authentication information is exploited to interact with the configured URL according to the defined flow. This way, the validation framework plays the role of both urban-data and urban-service providers by emulating a virtual user that interacts with the *SynchroniCity* instance, covering most of the possible actions that an actual user may perform. As a result, the validation framework creates and stores reports, for each instance, which include whether or not the different functionalities have worked as expected. Eventually, the reports can be accessed through an API to analyze the level of compliance of each city and its data. It is worth noting that all the APIs defined by *SynchroniCity* are Hyper-Text Transfer Protol (HTTP) based, so that in many cases the well-known HTTP response codes are used to test the correct behavior of the different interfaces.

As can be observed in Fig. 2, apart from the aforementioned authentication, the validation flow analyzes the Context and Storage Management APIs. In addition, the data exposed by each city is analyzed against the adopted data models. In the following, each of the validation flows is

[5] https://business-api-ecosystem.readthedocs.io/en/latest

[6] https://gitlab.com/synchronicity-iot/rz-instance-validator

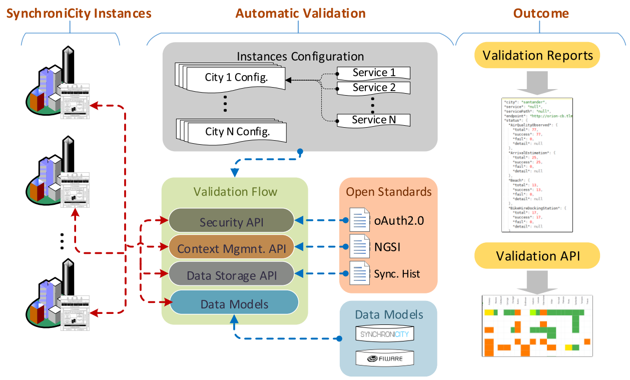

FIGURE 2. **Conceptual representation of the validation tool and flow.**

thoroughly described, as well as the information stored in the corresponding report.

A. SECURITY API

As mentioned before, OAuth 2.0 standard is adopted as the authorization solution within the implemented platform. In order to mimic the behavior of a user, the validation of the authorization API is performed in two steps: fist, an explicit validation, and then an implicit one. The explicit validation consists of obtaining an authorization token following one of the authorization flows defined in OAuth 2.0. To this end, for each pair city/service (see Fig. 2), the validation framework stores information of the configured validation flow, and the required credentials information. After that, the obtained access token is used to continue the validation of the data and other APIs, thus assessing the functionality of the authorization in an implicit manner. In particular the SynchroniCity instances must implement one of the following authentication flows [55]: *Authorization code grant*, *Implicit grant*, *Resource owner password credentials grant* or *Client credential grant*.

Eventually, the security API is considered as valid if the following happens: an access token can be obtained using the configured authorization flow, and no authorization error response (401 *unauthorized*) is obtained in the subsequent validation routines. Otherwise, if one or more of the aforementioned conditions do not hold, the validation of the security API fails, and the validation framework does not continue with the rest of the validation flows.

B. CONTEXT MANAGEMENT API

In this case, the framework checks that a given end-point is able to perform the following actions:

- *Context entity registration*: allows registering and updating of existing context entities.
- *Context entity search*: allows the discovery and retrieval of the context entities.
- *Context entity update*: allows updating context entities, related attributes and metadata. In addition, the end-point must be able to perform the following updates:
    - *append*: add or update attributes in an existing entity, otherwise the entity is created with the given attributes.
    - *update*: update attributes in an existing entity.
    - *delete*: remove attributes from a specified entity. If no attribute is indicated, it will remove the entity context information.
- *Context entity retrieval*: gathering of context information related to one or a list of entities.
- *Context entity subscription*: asynchronous notifications of changes about context information, based on multiple criteria over context attributes.

The validation of the Context Management API is carried out performing the most relevant queries defined by the NGSI specification. In order to better describe the different actions applied over the API, in Fig. 3a we present the test entity used for the validation. This entity also allows us to exemplify the type of data handled by the CDM module. As can be seen, an entity is a JavaScript Object Notation (JSON) document defined by an a duple type/identifier and embracing a number of attributes, which correspond to JSON properties. In turn, each attribute has a type and value. The former merely indicates the attribute type, while the latter can store any kind of value, ranging from numbers to other JSON documents. By using a test entity we know *a priori* the expected API behavior, so that validation conditions can be defined. Otherwise, the API response would be less predictable and defining a unified validation criteria would be more complex.

In Table 3 we indicate the different queries performed to validate the CDM, as well as the validation condition. First, the framework emulates an urban-data provider that interacts with the platform using the southbound interface. For that, an entity is created, and updated. Afterwards, playing the role of an urban-service provider that consumes data through the northbound interface, the framework retrieves the created entity using different NGSI methods and searching criteria. In addition, the validation flow also creates and deletes a subscription for changes in the test entity. Finally, the framework takes up the urban-data provider role to delete the test entity.

Eventually, the framework generates a CDM validation report that indicates whether or not each of the methods have been checked. It is worth noting that, if the entity creation is not successful, all the steps in Table 3 are marked as failure in the report.

C. DATA STORAGE API

Differently to the previous case, the DSM validation does not only rely on the API functionality, but also on existing data. This validation flow is configured with information about one entity for which historical information has been stored. It includes the entity identifier, type and the name of one of its attributes. Then the framework performs a set of routines to ensure that the API performs the pagination and temporal queries as expected. To illustrate the explanation, Fig. 3b depicts an example of a JSON document returned by the DSM

TABLE 3. Summary of the Context data Management APIs validation.

	Name	Method	URL	Validation condition	Comments
Southbound	Create Entity	POST	/v2/entities	Status code 201	
	Update & Check	POST GET	/v2/entities/{id}/attrs /v2/entities/{id}	Status code 204 Status code 200 and check that values are updated	It equals the value of `attribute1` to 8
	Delete Entity	DELETE	/v2/entities/{id}	Status Code 204	
Northbound	Query Entity	GET	/v2/entities/{id}?type={type}	Status Code 200	Type `test` is used for the query. Attributes' values are also validated
	Query Entity Attribute	GET	/v2/entities/{id}?type={type}& attrs={attr}	Status Code 200	Type `test` is used for the query. Attribute value is also validated
	Query Entity Attr. value	GET	/v2/entities/{id}?type={type}& attrs={attr}&options=values	Status Code 200	Type `test` is used for the query. Attribute value is also validated
	Find by Type	GET	/v2/entities?type={type}	Status Code 200	Type `test` is used for the query. It also validates that only one entity is returned and the attributes' values
	Find by Id pattern	GET	/v2/entities?type={type}& idPattern={pattern}	Status Code 200	Id pattern is `sc:apivalidation:*` and type `test` is used for the query. It also validates that only one entity is returned and the attributes' values
	Find by attr. value	GET	/v2/entities?type={type}&q= {attr}>1	Status Code 200	Type `test` is used for the query. It also validates that only one entity is returned and the attributes' values
	Create Subscription	POST	/v2/subscription	Status Code 201	It returns the subscription ID, used afterwards to validate subscription deletion
	Delete Subscription	DELETE	/v2/subscription/{subId}	Status Code 204	

```
{
  id: "sc:apivalidation:entity",
  type: "test",
  attribute1: {
    value: 10,
    type: "Float"
  },
  attribute2: {
    value: "String attribute",
    type: "String"
  }
}
```

(a) Test entity to perform NGSI validation

```
{
  id: "sc:apivalidation:entity",
  type: "test",
  attribute1: [
    {
      type: "Float",
      value: 10,
      modifiedAt: "2020-02-09T13:51:25.567"
    }, {
      type: "Float",
      value: 11,
      modifiedAt: "2020-02-09T13:51:28.304"
    },...
  ]
}
```

(b) Response of the historical API

FIGURE 3. Example of JSON documents handled by the CDM and DSM APIs.

API. As can be observed, it indicates the concerned entity, and a list of values of the requested attribute, along with the update value time. Thus, according to the concrete query the range of values varies.

In Fig. 4 we define 6 of the implemented routines, which exploit temporal or paging methods exposed by the API. Although other routines have been implemented, they are variations of the ones presented here, in which the order of the returned elements is swapped (ascent/descent). In particular, the routines have been defined to assert that the API exposed by *SynchroniCity* instance has a logical behavior and that equivalent queries, although with different parameters, behave similarly. However, we reckon that more complex and elaborated routines may be defined, and it is left for future framework extensions, along with the support for NGSI-LD historical API. In all cases, the routines start by querying the first (f) and last (l) record of the concerned attribute (*First&Last*() function). Afterwards, the queries make use of the first and last values as references. The routines implemented behave as follows:

- *Last samples after temporal reference*: as can be seen in Fig. 4a, this routine queries the last n samples after two temporal references. First, it uses the first value as references and checks that the query returns n elements and that the last one is equal to l. Afterwards, the last value l is used as reference, and the framework verifies that only the last value is returned.
- *Last samples before temporal reference*: this routine is described in Fig. 4b. After obtaining the first and last values, it queries the last n elements before the last value l, and checks that the returned data has n elements and the last one is effectively equal to l. After that, it queries the last n values before f and asserts that only one element is returned and it corresponds to f.
- *Last samples between temporal references*: as can be observed in Fig. 4c, this routine is simpler than the

```
$[f, l] \leftarrow First\&Last()$
$d \leftarrow GetLastAfter(n, f)$
$s \leftarrow size(d)$
if $s \neq n \,||\, d[n] \neq l$ then
    $return(Error)$
$d \leftarrow GetLastAfter(n, l)$
if $d \neq l$ then
    $return(Error)$
$return(Success)$
```

(a) Time after

```
$[f, l] \leftarrow First\&Last()$
$d \leftarrow GetLastBefore(n, l)$
$s \leftarrow size(d)$
if $s \neq n \,||\, d[n] \neq l$ then
    $return(Error)$
$d \leftarrow GetLastBefore(n, f)$
if $d \neq f$ then
    $return(Error)$
$return(Success)$
```

(b) Time before

```
$[f, l] \leftarrow First\&Last()$
$d \leftarrow GetLastBetween(n, f, l)$
$s \leftarrow size(d)$
if $s \neq n \,||\, d[n] \neq l$ then
    $return(Error)$
$return(Success)$
```

(c) Time between

```
$[f, l] \leftarrow First\&Last()$
$d_1 \leftarrow GetAfter(n, f, 0)$          ▷ Offset 0
$s \leftarrow size(d)$
if $s \neq n \,||\, d_1[0] \neq f$ then
    $return(Error)$
$d_2 \leftarrow GetAfter(n, f, 1)$          ▷ Offset 1
if $d_2[0] \neq d_1[1]$ then
    $return(Error)$
$return(Success)$
```

(d) Paging after

```
$[f, l] \leftarrow First\&Last()$
$d_1 \leftarrow GetBefore(n, l, 0)$         ▷ Offset 0
$s \leftarrow size(d)$
if $s \neq n \,||\, d_1[0] \neq f$ then
    $return(Error)$
$d_2 \leftarrow GetBefore(n, l, 1)$         ▷ Offset 1
if $d_2[0] \neq d_1[1]$ then
    $return(Error)$
$return(Success)$
```

(e) Paging before

```
$[f, l] \leftarrow First\&Last()$
$d_1 \leftarrow GetBetween(n, f, l, 0)$     ▷ Offset 0
$s \leftarrow size(d)$
if $s \neq n \,||\, d_1[0] \neq f$ then
    $return(Error)$
$d_2 \leftarrow GetBetween(n, f, l, 1)$     ▷ Offset 1
if $d_2[0] \neq d_1[1]$ then
    $return(Error)$
$return(Success)$
```

(f) Paging between

FIGURE 4. Logic of each routine defined for the DSM API validation.

previous ones. It queries the last n values between f and l, and checks that the returned data has the requested number of elements and the last one is equal to l.

- *Paging after reference*: as depicted in Fig. 4d, in this routine the n after f are queried with offsets 0 and 1. Then, it is checked that the second element with offset 0 is equal than the first one with offset 1. Thus checking the logical API behavior.
- *Paging before reference*: as shown in Fig. 4e, in this case the framework queries the same data as before, but in this case l is used as reference value. Again, the routine checks that the second element obtained with offset 0 is equal to the first one returned with offset 1.
- *Paging between references*: in this routine the framework uses both f and l as references and queries the values in between, as shown in Fig. 4f. Again, in practice, the query returns the same elements as before and the same validation is performed.

D. DATA MODELS

Apart from the APIs validation the framework can be configured to perform periodic validation of the data exposed by each *SynchroniCity* instance. In this sense, the defined data models set syntactic and semantic requirements over the representation of urban data. Context entities are categorized according to the urban service they belong to, and classified by types (e.g. urban mobility service has entities of type `BusStop`). Data models are defined for each entity type, and the tool checks that all the entities 1) have the required attributes, 2) the attributes are properly formatted, and 3) the attribute value is meaningful (e.g. the temperature cannot be negative).

First, the validation framework queries all the entity types exposed by each NGSI API. Afterwards, for each type, all the entities are retrieved and checked against the adopted data models. Considering that the data model definition can vary over time, the framework uses the information about the adopted data models in the *SynchroniCity* repository.[7] Then, it clones the *SynchroniCity* repository itself and the FIWARE's[8] one to use the datamodels definition (JSON schemas) during the validation. Under the veil, the validation framework uses the popular schema validation engine *AJV*[9] to carry out the actual validation against the defined schemas.

Since the number of entities can be very high, any lack of compliance when creating a type of entities would draw an intractable amount of error messages. In order to make the validation reports more usable, in case of data-model validation failure the report only contains a hint embracing detailed information of the first 5 context entities that are not compliant.

As an example, in Fig. 5 we show a fragment of the data model validation result that includes information about successful and failing validation. As can be observed in

[7] https://gitlab.com/synchronicity-iot/synchronicity-data-models.git

[8] https://github.com/Fiware/dataModels.git

[9] https://ajv.js.org/keywords.html

```
{
  "city" : "santander",
  "service" : "null",
  "servicePath" : "null",
  "endpoint" : "http://orion-cb.tlmat.
      synchronicity-iot.eu:1026/v2",
  "status" : {
    "AirQualityObserved" : {
      "total" : 77,
      "success" : 77,
      "fail" : 0,
      "detail" : null
    },
    "WeatherForecast" : {
      "total" : 6,
      "success" : 3,
      "fail" : 3,
      "detail" : [
        {
          "status" : "ENTITY_ERROR",
          "entityId" : "urn:ngsi-ld:
              WeatherForecast:santander:
              CurrentSlot",
          "detail" : [{
            "dataPath" : ".windDirection",
            "params" : {
              "comparison" : ">=",
              "limit" : 0,
              "exclusive" : false
            },
            "message" : "should be >= 0"
          }
        ]
      },
      ...
    },
    "lastUpdate" : ISODate("2020-09-23T09:45:27.49
        6+02:00")
}
```

FIGURE 5. Example of data models validation over the default service of Santander SynchroniCity instance.

the figure, the validation report includes the endpoint over which it is performed, as well as the city and service. Then, for each entity type, it indicates the total number of entities and how many of them are compliant with the data model definition. In the example, there are 77 valid entities of type `AirQualityObserved` and 6 entities of type `WeatherForecast`, not all of which are fully compliant. When errors are detected, the validation report stores the message returned by the validation engine, which provides detailed information about the concrete mistake.

V. PILOTS AND VALIDATION

As commented before, the validation framework has been used to ensure that the different *SynchroniCity* instances adopt the defined MIM. In order to illustrate the applicability of the framework, in this section we describe the main outcomes of the validation carried out during the open call. First, we will describe the open call and the validation process. Later on, the main results will be presented.

A. OPEN CALL PROCESS

The open-call was open to cities worldwide for 6 months. The rules and requirements were specifically tailored for Small and Medium Enterprises (SMEs), while not excluding large businesses and cities willing to participate in support of SMEs as lead applicants. The aim of the open call was to provide real-life solutions to validate the *SynchroniCity* architecture regarding service interoperability and replicability.

TABLE 4. Cities of deployment by Pilot Group.

	Reference Zones								
Pilot	Antwerp	Carouge	Eindhoven	Helsinki	Manchester	Milano	Porto	Santander	New cities
Active Travel	■			■	■				
ASAP value		■							Bordeaux, Seongnam
Aut. Hub							■		Donegal, La Nucía, Bezana, Palencia
AAQM				■			■		Tampere
Blue Alpaca	■			■		■		■	
CASD	■	■		■					
Enc. Cycling	■				■				Dublin
Kimap City						■	■	■	
Kissmybike	■					■		■	
LINC		■				■	■		
Neighbourly					■		■	■	Herning
NoiseAbility			■						Bilbao, Edinburgh
QUAMTRA		■					■		Calatayud
Rain Brain	■		■						
RealTime Traffic	■						■		
Smart Impact		■						■	Novi Sad

To this end, among the requirements for open call eligibility, pilots were to propose a solution that would be deployed in at least 2 cities and would tackle challenges in one of three specific theme areas (mobility, citizen engagement, environment and wellbeing) and one general (open theme). During the application period more than 130 applications were received, from both single SMEs or full consortia (including new cities), and involving the RZs. Pilots were submitted by lead applicants from more than 20 countries.

The most popular challenges were the ones defined by the open-call, followed by decreasing air and noise pollution, mobility-as-a-service and citizen involvement in decision making. The massive response to the open call allowed the selection of 16 high profile pilots, deploying 50 IoT enabled services in 21 cities delivered by 36 companies in 6 months. Table 4 summarizes the cities of deployment for each of the selected pilot groups. In the final column, it is indicated the new cities where the pilots were also deployed, in some cases the cities were part of the pilot consortium.

During the open call, the support structure surrounding the pilots and the cities was carefully considered. As the 6-month pilot period was relatively short, each SME was assigned a mentor to help them navigate the different stakeholders and escalate issues to the necessary parties. These mentors were representatives of organizations that were part of the *SynchroniCity* consortium and were the main point of contact for any issue the pilot may have had during the pilot project (e.g. general technical issues, reporting and admin issues, generic issues with the cities, etc.). In addition, a helpdesk was set up as a point of contact for the *SynchroniCity* consortium to support pilots in general issues (administrative, financial, legal, ethical) and, exceptionally, they also make technical inquiries that could not be solved by the mentors.

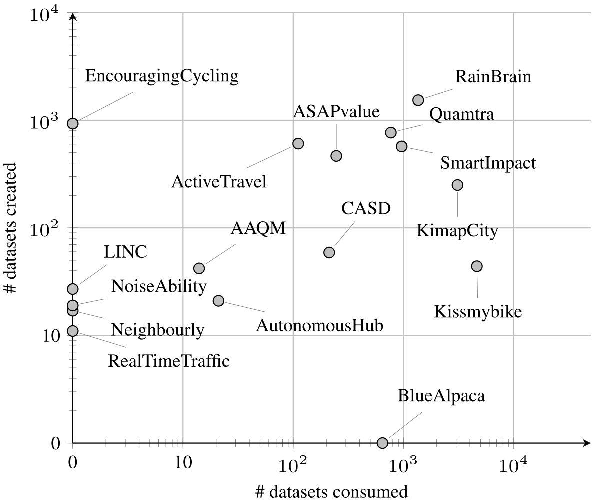

FIGURE 6. Number of datasets consumed and created by each pilot.

B. MAIN VALIDATION RESULTS

The success of open call lies in the fact that all 16 pilot groups managed to deploy their solutions across a minimum of two cities, ensuring integration and replicability, within the stringent open call period. In total, the pilot groups deployed 37 solutions over *SynchroniCity* instances. Apart from the 8 RZs, the architecture was replicated in 13 new cities, leading to a total number of 21 instances. The pilot groups engaged a wide group of stakeholders, including city departments, citizens, schools and third-party service providers, so ensuring many areas of society were involved in the deployment of the IoT solutions. In the following we summarize the main results drawn by the open call.[10]

1) DATASETS USED

All 16 pilots achieved the minimum level of technology integration required. This included integration with the *SynchroniCity* architecture being used within their solution, as well as datasets being created and/or consumed using the deployed instances.

First, in Fig. 6 we represent (in log scale) the number of entities consumed and created by each of the pilot groups. As can be observed, most of the pilots created and consumed data in a comparable scale. On the other hand, we can also observe that in some cases the deployed services were only either data providers or consumers. Concretely, only one pilot (over the X axis) was a pure data consumer, while a number of them (over the Y axis) were only data providers.

As for datasets, as commented before, a total of 50 data models were adopted or defined. According to these data-models 159 datasets were generated belonging to different verticals. Fig. 7 showcases the percentage of generated datasets that belong to each of the vertical or data category, previously enumerated in Table 2. As can be observed, the number of data models of each category and datasets generated are not balanced. In this sense, more than 55% of the datasets belong to *Point of Interest* and *Park and Gardens* category, while only 6 data-models out of 50 were defined for these verticals. Opposite to that, *Transportation* and *Mobility* group 27 data models, but just the 3.24% of the generated datasets.

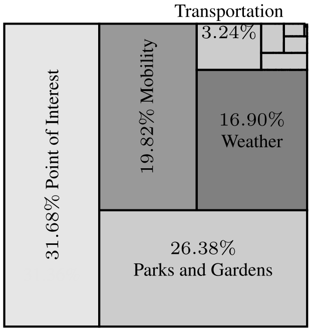

FIGURE 7. Distribution of datasets by service vertical.

The reasons for this unbalance in many-fold. On the one hand, the possibility to access different types of information is not uniform. While tourism information (i.e. points of interest) is easily accessible, accessing to information belonging to companies or urban facilities (i.e. transportation) requires complex procedures. On the other hand, the amount of information included in the data models is also unequal. Some data models store information about single places or buildings, while others have complex information models to represent systems.

During the open call, the validation framework periodically checked the generated data. Apart from the access to the validation reports, a live web page (`https://validation.services.synchronicity-iot.eu/table/`) was created for helping the pilots and new cities to know the status of the data. Altogether, the validation framework checked for than 400, 000 entities generated in all the instances.

2) IoT SOLUTIONS DEPLOYED

Then, in Fig. 8 we represent the number of times pilots' services have been replicated in different cities. Altogether, 37 services have been deployed in the RZs, while 13 in the new cities. As can be observed, most of the pilots have deployed their solution in at least 3 cities, and half of them opted for new cities (different to RZs) for their services.

3) TECHNICAL INTEGRATION

All pilots were technically approved but there were differing levels of integration with *SynchroniCity*. In order to provide a general view of the integration, three levels were defined according to the previous results:

- Minimum integration: it fulfills the open call minimum integration requirements, in the sense that they just generate datasets in at least 2 pilot cities. However, the

[10] Detailed descriptions of pilots and concrete SynchroniCity instances can be found in the following link: `https://synchronicity-iot.eu/cities-pilots/`

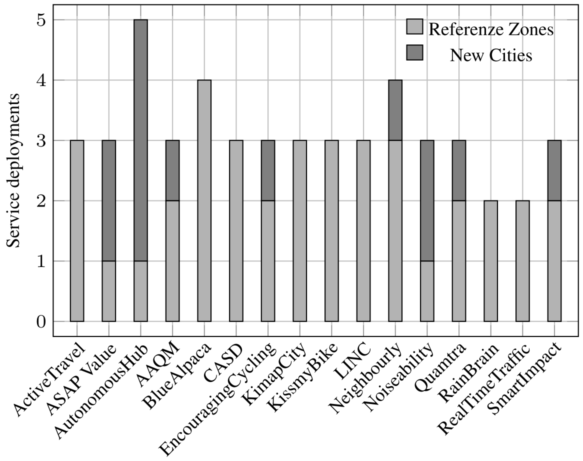

FIGURE 8. Number of instances where pilots were deployed.

solution does not have a strong dependency on *SynchroniCity*.

- Medium integration: the performance of the solution depends on *SynchroniCity*, as it consumes data from instances. In this sense, the solution requires the *SynchroniCity* APIs and data-models to work.
- High integration: it has the same dependency as the medium integration, and in addition the pilot has brought new cities were the *SynchroniCity* architecture has been deployed.

Based on the above criteria, Fig. 9 shows the overall integration level reached during the open call. As can be seen, 19% of the pilots (3 out of 16) have achieved a high integration level. On the other hand, almost one third of the pilots (5 pilots) just fulfilled the minimum open call demandable requirements. Finally, in the 50% of cases, the pilots achieved a medium integration level, making the *SynchroniCity* instances a relevant part of their solution.

All in all, more than two thirds of the services deployed by pilots have been strongly integrated with *SynchroniCity* in several cities, endorsing the replication and integration capabilities of the platform. In addition, considering the open call duration (6 months), the achieved integration level demonstrate the adopted MIMs were easily included in existing solutions. In this sense, the periodic validation flow over both APIs and data models ensured that the different instances presented a predictable behavior, enabling an agile deployment of services.

C. LESSONS LEARNED

In this section, we enumerate some lessons learned from the feedback provided by pilots' SMEs, as well as from observations done by mentors and during the technical support.

1) STANDARDS

Despite some progress, it is required to continue elaborating the data modeling and API convergence to enable smart cities interoperability. SMEs are often at the cutting edge or emerging standards and frameworks, pushing development in the real world, often ahead standards. Establishing bridges between standard bodies and SMEs would help to make standards evolve faster. In this sense, SMEs should engage standardization bodies at so early stage to identify relevant standards, raise issues where there is a lack of convergence between definition and actual needs, and define needs for new standards.

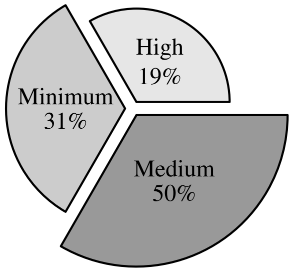

FIGURE 9. Pilot level of integration to SynchroniCity instances.

2) TECHNICAL ASPECTS

Even if common standards are used, there are some differences regarding the setup and available options. In particular, some pilots groups pointed out the differences in the authentication process and options available in the different cities, which boils down to different OAuth 2.0 authorization flows availability.

In addition, the data isolation strategy in the different cities has also led to slight differences in the way pilots had to tackle the integration. Some cities have made use of the multi-tenancy NGSI feature, so that a single endpoint is used for all the pilots for both consuming and creating data. However, in order to simplify the integration of legacy frameworks with the *SynchroniCity* one, other cities decided to deploy multiple context broker instances for each pilot to consume and create their data, while a city-wide endpoint was provided to consume city data. Although in all cases NGSI is used as context API, in the second option pilots are required to differentiate between city and pilot data, making the integration more complex.

3) CITY AND SME COORDINATION

Every city is different when creating the relationship with the relevant stakeholders, including service providers. Considering those differences each city requires a slightly different approach to both accessing contextual datasets to use by SMEs, as well as developing project use cases. An agile approach for the management of the individual projects needs to be adopted, which places further importance on early stage project development, teasing out specific and achievable reports for cities to ensure useful data generation.

To this end, it is necessary that cities define agile and clear internal procedures, embracing administrative, legal and technical aspects. In this sense, pilots highlighted the need for the cities to define technical contacts points early on, and coordination strategies.

VI. CONCLUSION

Due to the maturity reached by the IoT technologies and its pervasive presence, its adoption is now a fundamental aspect to evolve urban environments. Although many solutions and approaches have been postulated in the last years, it is still needed to develop the means to further evolve the IoT based services in urban ecosystems. It embraces the easy replication of services to help businesses of all sizes to scale up their products.

Considering the above, the *SynchroniCity* framework postulates as an easy to adopt solution to fulfill the aforementioned requirements. Opposed to clean-slate solutions, the framework has been co-designed with cities and service providers, to ensure that its adoption on top of existing solutions is feasible. In addition, it is defined based on open-standards foster a neutral architecture, thus opening the urban services scenario to business of different sizes avoiding vendor lock-in effect. Meanwhile framework definitions set the basis for service replicability, it is necessary to practically ensure that exposed APIs and data follow the adopted standards and definitions. This way, it is ensured that smart city ecosystems present predictable and uniform interfaces, so that urban stakeholders do not need to develop customized solutions for each city. To this end, in this work we have presented a validation framework to ensure compliance with the *SynchroniCity* framework. The framework has been extensively exploited during the open call carried out by the *SynchroniCity* consortium to replicate the framework in a total of 21 cities worldwide.

According to the results, we can asseverate that the framework has been easily adopted by cities, even by those with existing smart city solutions. Similarly, the results yielded by the open call evince that SMEs, or open call pilots, have been able easily adapt their existing services to integrate with *SynchroniCity*, which allows their solutions to be replicated in the different cities. We understand that the continuous validation process has played a fundamental role in this aspect. Service providers rely on the fact that all cities interfaces and data had been validated, so that service replication could be done smoothly.

However, we have also observed that the level of integration is unequal among the pilots. While in most cases the integration degree has led to new solutions dependent of the *SynchroniCity*, in some cases (30%) such integration is low. Furthermore, we have analyzed how pilots' services interact with the framework in terms of data generation and consumption. According to the results, most of the services consume and generate data, but in a few cases the data is only generated. In addition, there was one pilot, which was a pure data consumer.

Although the *SynchroniCity* project is over, its outcomes are being further developed, as commented in [56]. In this sense, the systematic validation tools described in this work play a relevant role to foster replicability and reusability.As for future work, we will consolidate these tools by adopting more data models and standards. In particular, the validation framework is being adapted to use the data models defined by the Smart Data Models initiative.[11] In addition, we ar extending the APIs validation flows to cover more NGSI functionalities and include NGSI-LD compatibility. On the other hand, we are working with FIWARE Foundation[12] in isolating some functionalities of the validation framework to develop middleware able to perform data model validation on the fly. This way, the validation will take place within the datapath, thus ensuring that stored data is always compliant with the adopted data models.

ACKNOWLEDGMENT

The content of this paper does not reflect the official opinion of the European Union. Responsibility for the information and views expressed therein lies entirely with the authors.

REFERENCES

[1] G. D'Amico, P. L'Abbate, W. Liao, T. Yigitcanlar, and G. Ioppolo, ''Understanding sensor cities: Insights from technology giant company driven smart urbanism practices,'' *Sensors*, vol. 20, no. 16, p. 4391, Aug. 2020, doi: 10.3390/s20164391.

[2] T. Yigitcanlar, L. Butler, E. Windle, K. C. Desouza, R. Mehmood, and J. M. Corchado, ''Can building 'artificially intelligent cities' safeguard humanity from natural disasters, pandemics, and other catastrophes? An urban scholar's perspective,'' *Sensors*, vol. 20, no. 10, p. 2988, May 2020, doi: 10.3390/s20102988.

[3] T. Yigitcanlar, K. C. Desouza, L. Butler, and F. Roozkhosh, ''Contributions and risks of artificial intelligence (AI) in building smarter cities: Insights from a systematic review of the literature,'' *Energies*, vol. 13, no. 6, p. 1473, Mar. 2020, doi: 10.3390/en13061473.

[4] T. Yigitcanlar and F. Cugurullo, ''The sustainability of artificial intelligence: An urbanistic viewpoint from the lens of smart and sustainable cities,'' *Sustainability*, vol. 12, no. 20, p. 8548, Oct. 2020, doi: 10.3390/su12208548.

[5] H. Habibzadeh, C. Kaptan, T. Soyata, B. Kantarci, and A. Boukerche, ''Smart city system design: A comprehensive study of the application and data planes,'' *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–38, May 2019. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3309545

[6] A. Antonić, M. Marjanović, and I. P. Žarko, ''Modeling aggregate input load of interoperable smart city services,'' in *Proc. 11th ACM Int. Conf. Distrib. Event-Based Syst. (DEBS)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 34–43. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3093742.3093928

[7] D. Kramer and J. Haehner, ''Beyond semantic interoperability in IoT ecosystems,'' in *Proc. 8th Int. Conf. Internet Things (IOT)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–4. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3277593.3277639

[8] B. Ahlgren, M. Hidell, and E. C.-H. Ngai, ''Internet of Things for smart cities: Interoperability and open data,'' *IEEE Internet Comput.*, vol. 20, no. 6, pp. 52–56, Nov. 2016.

[9] M. Pradhan, ''Interoperability for disaster relief operations in smart city environments,'' in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, Apr. 2019, pp. 711–714.

[10] A. Taamallah, M. Khemaja, and S. Faiz, ''Toward a framework for smart city strategies design,'' in *Proc. 3rd Int. Conf. Smart City Appl. (SCA)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–7. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3286606.3286833

[11] `https://www.fiware.org/developers/smart-data-models/`

[12] `https://www.fiware.org/foundation/`

[11] F. Lopes, S. Loss, A. Mendes, T. Batista, and R. Lea, "SoS-centric middleware services for interoperability in smart cities systems," in *Proc. 2nd Int. Workshop Smart (SmartCities)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1–6. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3009912.3009917

[12] R. Lea and M. Blackstock, "Smart cities: An IoT-centric approach," in *Proc. Int. Workshop Web Intell. Smart Sens. (IWWISS)*. New York, NY, USA: Association for Computing Machinery, 2014, pp. 1–2, doi: 10.1145/2637064.2637096.

[13] D. Makoshenko and I. Enkovich, "IoT development: Discovering, enabling and validation of real life IoT scenarios," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2017, pp. 159–164.

[14] ITU-SG5, "Information and communication technologies for climate change adaptation in cities," Tech. Rep. Accessed: Dec. 21, 2020. [Online]. Available: https://www.itu.int/md/T13-SG05-151012-TD-GEN-1007

[15] ISO/IEC JTC1. *Smart Cities. Preliminary Report 2014*. Accessed: Dec. 21, 2020. [Online]. Available: https://www.iso.org/files/live/sites/isoorg/files/developing_standards/docs/en/smart_cities_report-jtc1.pdf

[16] AIOTI, "AIOTI strategy 2017–2021," Tech. Rep. Accessed: Dec. 21, 2020. [Online]. Available: https://aioti.eu/wp-content/uploads/2017/11/AIOTI_Stategy_2017-2021_V1.0_FINAL_WEB.pdf

[17] ITU-SG20. *Focus Group on Data Processing and Management to Support IoT and Smart Cities & Communities*. Accessed: Dec. 21, 2020. [Online]. Available: https://www.itu.int/en/ITUT/focusgroups/dpm/Pages/default.aspx

[18] U4SSC, "United for smart sustainable cities—Overview," Tech. Rep. Accessed: Dec. 21, 2020. [Online]. Available: https://www.unece.org/fileadmin/DAM/hlm/projects/SMART_CITIES/U4SSCbrochure.pdf

[19] S. Soursos and I. P. Zarko, "SymbIoTe: Symbiosis of smart objects across IoT environments," in *Digitising the Industry—Internet of Things Connecting the Physical, Digital and Virtual Worlds*, IERC Cluster Book, Ed. Gistrup, Denmark: River Publishers, 2016, pp. 303–307.

[20] W. Vandenberghe, B. Vermeulen, P. Demeester, A. Willner, S. Papavassiliou, A. Gavras, M. Sioutis, A. Quereilhac, Y. Al-Hazmi, F. Lobillo, F. Schreiner, C. Velayos, A. Vico-Oton, G. Androulidakis, C. Papagianni, O. Ntofon, and M. Boniface, "Architecture for the heterogeneous federation of future Internet experimentation facilities," in *Proc. Future Netw. Mobile Summit*, Jul. 2013, pp. 1–11.

[21] M. Serrano *et al.*, "Cross-domain interoperability using federated interoperable semantic IoT/cloud testbeds and applications: The FIESTA-IoT approach," in *Building the Future Internet through FIRE: 2016 Fire Book, a Research and Experimentation Based Approach*. Gistrup, Denmark: River Publishers, 2017, pp. 287–321.

[22] V. Gutiérrez, E. Theodoridis, G. Mylonas, F. Shi, U. Adeel, L. Diez, D. Amaxilatis, J. Choque, G. Camprodom, J. McCann, and L. Muñoz, "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

[23] D. Amaxilatis, D. Boldt, J. Choque, L. Diez, E. Gandrille, S. Kartakis, G. Mylonas, and L. S. Vestergaard, "Advancing experimentation-as-a-service through urban IoT experiments," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2563–2572, Sep. 2019.

[24] J. B. Bernabe, I. Elicegui, E. Gandrille, N. Gligoric, A. Gluhak, C. Hennebert, J. L. Hernandez-Ramos, C. López, A. Manchinu, K. Möessner, M. Nati, C. O'Reilly, N. Palaghias, A. Pintus, L. Sánchez, A. Serra, and R. van Kranenburg, "SocIoTal—The development and architecture of a social IoT framework," in *Proc. Global Internet Things Summit (GIoTS)*, Jun. 2017, pp. 1–6.

[25] T. Akiyama, S. Murata, K. Tsuchiya, T. Yokoyama, M. Maggio, G. Ciulla, J. R. Santana, M. Zhao, J. B. D. Nascimento, and L. Gürgen, "FESTIVAL: Design and implementation of federated interoperable smart ICT services development and testing platform," *J. Inf. Process.*, vol. 25, pp. 278–287, Mar. 2017.

[26] J.-P. Exner, "The ESPRESSO—Project—A European approach for smart city standards," in *Computational Science and Its Applications—ICCSA*. Cham, Switzerland: Springer, 2016, pp. 483–490.

[27] *The European Innovation Partnership on Smart Cities and Communities EIP-SCC*. Accessed: Dec. 21, 2020. [Online]. Available: https://eu-smartcities.eu/page/about

[28] BIG IoT Project. *Bridging the Interoperability Gap of the Internet of Things*. Accessed: Dec. 21, 2020. [Online]. Available: http://big-iot.eu/

[29] E. Kovacs, M. Bauer, J. Kim, J. Yun, F. L. Gall, and M. Zhao, "Standards-based worldwide semantic interoperability for IoT," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 40–46, Dec. 2016.

[30] A. Sharma, S. Acharya, V. Rajaraman, R. Ramesh, A. Babu, and B. Amrutur, "Schemas for IoT interoperability for smart cities," in *Proc. 4th ACM Int. Conf. Syst. Energy-Efficient Built Environ. (BuildSys)*. New York, NY, USA: Association for Computing Machinery, 2017 pp. 1–2. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3137133.3141466

[31] V. A. Barros, J. C. Estrella, L. B. Prates, and S. M. Bruschi, "An IoT-daas approach for the interoperability of heterogeneous sensor data sources," in *Proc. 21st ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst. (MSWIM)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 275–279. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3242102.3242141

[32] K. Chaturvedi and T. H. Kolbe, "Intersensor service: Establishing interoperability over heterogeneous sensor observations and platforms for smart cities," in *Proc. IEEE Int. Smart Cities Conf. (ISC)*, Sep. 2018, pp. 1–8.

[33] H.-L. Truong, L. Gao, and M. Hammerer, "Service architectures and dynamic solutions for interoperability of IoT, network functions and cloud resources," in *Proc. 12th Eur. Conf. Softw. Archit., Companion (ECSA)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–4. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3241403.3241407

[34] H. Kim, A. Ahmad, J. Hwang, H. Baqa, F. Le Gall, M. A. R. Ortega, and J. Song, "IoT-TaaS: Towards a prospective IoT testing framework," *IEEE Access*, vol. 6, pp. 15480–15493, 2018.

[35] S. D. Nagowah, H. B. Sta, and B. A. Gobin-Rahimbux, "Towards achieving semantic interoperability in an IoT-enabled smart campus," in *Proc. IEEE Int. Smart Cities Conf. (ISC)*, Oct. 2019, pp. 593–598.

[36] A. Gyrard, S. K. Datta, and C. Bonnet, "A survey and analysis of ontology-based software tools for semantic interoperability in iot and wot landscapes," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Feb. 2018, pp. 86–91.

[37] A. Mazayev, J. A. Martins, and N. Correia, "Interoperability in IoT through the semantic profiling of objects," *IEEE Access*, vol. 6, pp. 19379–19385, 2018.

[38] N. Villanueva-Rosales, L. Garnica-Chavira, V. M. Larios, L. Gómez, and E. Aceves, "Semantic-enhanced living labs for better interoperability of smart cities solutions," in *Proc. IEEE Int. Smart Cities Conf. (ISC)*, Sep. 2016, pp. 1–2.

[39] A. Gyrard and M. Serrano, "Connected smart cities: Interoperability with SEG 3.0 for the Internet of Things," in *Proc. 30th Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*, 2016, pp. 796–802.

[40] J. An, F. Le Gall, J. Kim, J. Yun, J. Hwang, M. Bauer, M. Zhao, and J. Song, "Toward global IoT-enabled smart cities interworking using adaptive semantic adapter," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5753–5765, Jun. 2019.

[41] N. Kolbe, S. Kubler, J. Robert, Y. Le Traon, and A. Zaslavsky, "Towards semantic interoperability in an open IoT ecosystem for connected vehicle services," in *Proc. Global Internet Things Summit (GIoTS)*, 2017, pp. 1–5.

[42] M. Blackstock and R. Lea, "Toward interoperability in a Web of things," in *Proc. ACM Conf. Pervas. Ubiquitous Comput. Adjunct Publication (UbiComp)*. New York, NY, USA: Association for Computing Machinery, 2013, pp. 1565–1574. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/2494091.2497591

[43] O. Novo and M. D. Francesco, "Semantic interoperability in the IoT: Extending the Web of things architecture," *ACM Trans. Internet Things*, vol. 1, no. 1, pp. 1–25, Mar. 2020. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/3375838

[44] J. L. Hernández, R. García, M. Fischer, J. Schonowski, D. Atlan, and T. Ruohomäki, "An interoperable open specifications framework for smart city urban platforms," in *Proc. Global IoT Summit (GIoTS)*, 2019, pp. 1–7.

[45] P. Grace, J. Barbosa, B. Pickering, and M. Surridge, "Taming the interoperability challenges of complex IoT systems," in *Proc. 1st ACM Workshop Middleware Context-Aware Appl. IoT (MIOT)*. New York, NY, USA: Association for Computing Machinery, 2014, p. 1–6. [Online]. Available: https://doi-org.unican.idm.oclc.org/10.1145/2676743.2676744

[46] A. Zabasta, N. Kunicina, K. Kondratjevs, A. Patlins, L. Ribickis, and J. Delsing, "MQTT service broker for enabling the interoperability of smart city systems," in *Proc. Energy Sustainability Small Developing Economies (ESDE)*, 2018, pp. 1–6.

[47] M. Blackstock and R. Lea, "IoT interoperability: A hub-based approach," in *Proc. Int. Conf. Internet Things (IOT)*, 2014, pp. 79–84.

[48] I. P. Žarko, S. Mueller, M. Płociennik, T. Rajtar, M. Jacoby, M. Pardi, G. Insolvibile, V. Glykantzis, A. Antonić, M. Kušek, and S. Soursos, "The symbiote solution for semantic and syntactic interoperability of cloud-based IoT platforms," in *Proc. Global IoT Summit (GIoTS)*, 2019, pp. 1–6.

[49] M. Pradhan, N. Suri, C. Fuchs, T. H. Bloebaum, and M. Marks, "Toward an architecture and data model to enable interoperability between federated mission networks and IoT-enabled smart city environments," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 163–169, Oct. 2018.

[50] SynchroniCity. (Jun. 2017). *Guidelines for Synchronicity Architecture*. [Online]. Available: https://synchronicity-iot.eu/media/

[51] SynchroniCity. (Aug. 2018). *Reference Architecture for IoT Enabled Smart, Cities, Updte*. [Online]. Available: https://synchronicity-iot.eu/media/

[52] Open Mobile Alliance (OMA), "NGSI context management," Tech. Rep. 1.0, May 2012. Accessed: Dec. 21, 2020. [Online]. Available: http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-AD-NGSI-V1_0-20120529-A.pdf

[53] ETSI GS CIM, "Context information management (CIM): NGSILD API," Tech. Rep. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf

[54] GSMA. (Oct. 2018). *IoT Big Data Harmonised Data Model*. [Online]. Available: https://www.gsma.com/iot/wp-content/uploads/2018/11/CLP.26-v6.0.pdf

[55] D. Hardt, *The OAuth 2.0 Authorization Framework*, document RFC 6749, Internet Requests for Comments, Oct. 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6749.txt

[56] F. Cirillo, D. Gómez, L. Diez, I. E. Maestro, T. B. J. Gilbert, and R. Akhavan, "Smart city IoT services creation through large scale collaboration," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5267–5275, Jun. 2020.

LUIS DIEZ received the M.Sc. and Ph.D. degrees from the University of Cantabria, in 2013 and 2018, respectively. He is currently an Assistant Professor with the Communications Engineering Department, University of Cantabria. He has been involved in different international and industrial research projects. His research focuses on future network architectures, resource management in wireless heterogeneous networks, and the IoT solutions and services. He has published more than 40 scientific and technical articles in those areas. He has served as a TPC member and a reviewer in a number of international conferences and journals. As for teaching, he has supervised 15 B.Sc. and M.Sc. theses. He teaches in courses related to cellular networks, network dimensioning, and service management.

JOHNNY CHOQUE received the degree in electronic engineering from the National University of Engineering, Peru, in 1995, and the Ph.D. degree *(cum laude)* in communications engineering from the University of Cantabria, Spain, in 2014. Since 2000, he has been employed as a Researcher with the University of Cantabria, working in several projects of Framework Programme and Horizon 2020 of EU. His current research interests include LPWAN, smart cities, the IoT, and blockchain.

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

LUIS MUÑOZ (Senior Member, IEEE) was born in Barcelona, Spain, in 1964. He received the Telecommunications Engineering and Ph.D. degrees from the Polytechnical University of Cataluña (UPC), Spain, in 1990 and 1995, respectively. He is currently the Head of the Network Planning and Mobile Communications, University of Cantabria, Spain. His research focuses on advanced data transmission techniques, heterogeneous wireless multihop networks, the Internet of Things, smart cities, and applied mathematical methods for telecommunications. He has participated in several National and European research projects belonging to the fourth, fifth, sixth, and seventh Framework Program, in which he was the Technical Manager of SmartSantander. He has published over 150 journal and conference papers. He serves as an editor of several journals and he has been invited to participate in the Steering Committee and Technical Program Committee of the most relevant international conferences. In parallel to this activity, he serves as a consultant for the Spanish Government as well as for different companies in Europe and USA. He has served as an expert of the ETSI and European Commission.