# Developing Complex Services in an IoT Ecosystem

Charilaos Akasiadis, Grigorios Tzortzis, Evaggelos Spyrou, and Constantine Spyropoulos

Institute of Informatics and Telecommunications
NCSR–"Demokritos"
Aghia Paraskevi, 15310, Greece
Email: {cakasiadis, gtzortzi, espyrou, costass}@iit.demokritos.gr

***Abstract*—Recent advancements in single board computers, communications technologies and protocols, as well as the concepts of service-oriented architectures (SoA) and everything as a service (EaaS), constitute a prelude to the Internet of Things (IoT) revolution. Billions of devices are interconnected and integrated as modular web services, which can be used and re-used by developers making the building and realization of complex applications easier. In this work, we take advantage of the SYNAISTHISI platform, which is able to interface and integrate devices, services and humans, and expose their capabilities as virtualized semantically annotated services that can be mashed into applications. We analyze the development process from a developer's perspective, present an ontology for smart meeting rooms and focus on a real-world case, that is delivering a complex application for counting the persons in the interior of a smart meeting room, using technologies that support IoT.**

***Keywords*-IoT platform; ontology; semantics; service discovery; service composition;**

## I. Introduction

During the last few years, everyday physical objects have been modified with the embodiment of short-range and energy efficient mobile transceivers and have been enhanced with unique identifiers. The networking of such objects, originally not meant to be "computerized", has led to the so-called "Internet-of-Things" (IoT) [1], which is considered as the next industrial revolution [2]. IoT is expected to find applications in many areas, such as industry, logistics, building and home automation, smart cities, smart manufacturing, healthcare, automotive, etc. The aforementioned are only a small fragment of the areas that benefit from IoT technologies [3], [4].

In most cases, IoT approaches follow a service-based architecture [5]. The provided services can be categorized into three distinctive yet interdependent types: *(i)* services that capture properties of the physical world and provide raw or slightly processed measurements (sensing services); *(ii)* services that process the acquired measurements and provide the inferred results (processing services); *(iii)* services that enable certain actions, based on these results (actuating services).

Thus, IoT applications involve distributed sensor networks at various scales, e.g. over the human body, small indoor areas (such as a house), or even world-scale outdoor areas. In any case, they are interconnected with other distributed services, possibly located on a cloud infrastructure. Their purpose is to process the measurements gathered and autonomously extract results, i.e. without the need of human intervention. Upon processing, some actuation elements may be triggered as well. Such services are offered to a variety of users and can be combined to provide other, more complex, services and applications. To achieve this, they must be easily discoverable and interconnectable by developers using the IoT ecosystem.

In this work, we describe the procedure of developing complex applications using an IoT compatible platform, named SYNAISTHISI [6]. In this platform, services are semantically enriched using ontologies. Specifically, we have implemented a smart meeting room ontology. We show how a developer may search for available services and the way simple sensing and processing services are interconnected to provide complex services, using existing, simpler ones. As a test case to illustrate our approach, we develop a "person counting" service, implemented within the context of a smart meeting room, that exploits a subset of the available sensing and processing services. The whole procedure becomes quite intuitive, without requiring the developer to have prior knowledge regarding the available services' implementation details (programming languages, technologies, etc.).

The rest of this paper is organized as follows: In Section II we provide an overview of related work. Then, in Section III we briefly present the SYNAISTHISI platform, the smart meeting room ontology, and discuss the steps required for service discovery and interconnection. In Section IV the use case of constructing a "person counting" service using existing, simpler services is presented. Finally, conclusions are drawn in Section V, where possible directions for future work are also discussed.

## II. Related Work

It is well-known that orchestrating a collective functionality using highly heterogeneous devices and modules, is quite a difficult task. There exist a few past attempts whose goal is to make complex applications' creation easier and simpler. For example, in [7], the authors combine various existing technologies to make applications using RFID more appealing. A recent paper [8] presents a suite of tools, which aim to aid the developer towards the creation of IoT applications. The work of [9] illustrates the process of service composition given IoT compatible components, in the context of transport applications, while [10], presents a case study of fleet monitoring.

In this work, we exploit the capabilities offered by the SYNAISTHISI platform [6] to *(i)* easily develop new, or

discover pre-existing modules, to use in a complex application, *(ii)* interconnect the inputs and outputs of the various services, and *(iii)* bring the complex application into realization and make it accessible from anywhere in the world, provided that internet connections and valid user accounts exist. We give descriptions of such efforts and try to get a glimpse of how development using IoT-enabling technologies will look like.

To offer developers a more natural and complete representation of services, usually, semantics are employed. Several ontologies exist for adding semantic content to web services, giving rise to the vision of semantic web services [11]. Two important frameworks towards this direction are OWL-S [12] and WSMO [13], which provide highly expressive models for describing web services. Such rich conceptual approaches, however, require considerable effort from users to comprehend and appropriately annotate the services, while reasoning over them constitutes a computationally intensive process. Another solution, the Minimal Service Model (MSM) [14], [15] captures the core semantics of web services, thus trading part of the expressivity of OWL-S and WSMO for efficiency and ease of use. A common feature of these approaches, however, is that they are general purpose upper ontologies, hence specializing them for domain-specific concepts demands additional effort. Therefore, we decided to develop an ontology that is tailored to the smart meeting room scenario.

A closely related attempt to the one we adopt in this work, can be found in [16] where BOnSAI, a smart building ontology, is introduced. In general, BOnSAI categorizes services/devices into two types, that is sensors and actuators, whereas in our case a third type, called processor, is also incorporated. Processors represent software components that are capable of executing various computations. The adoption of all three key service/device types makes our ontology able to adequately describe services found in a smart meeting room.

## III. Developing Applications Using an IoT-ready Platform

As it has already been stated in Section I, the advancement of the Internet of Things (IoT) technologies gave rise to service-oriented architectures (SoA). In this context, each sensor, processor, or actuator, is modeled as an online service, i.e. an S-, P-, or A-type service, respectively. Such services are discoverable and reachable over the internet via an IoT platform. This section describes the procedure of application development using an IoT-ready platform, called SYNAISTHISI, and the features that are used. From the developer's side, we assume that there exists some prior knowledge regarding service composition, i.e. which kinds of services are required and how they should be interconnected. Hence, the developer only needs to discover the services and define their communication routes.

### A. The SYNAISTHISI Platform

The goal of the SYNAISTHISI project[1] is to deliver energy efficient, secure, and effective applications and services to end users, which aim to minimize the environmental impact, monetary costs, user discomfort, delays and utilization of resources. In this section we provide a brief description of the architecture of the SYNAISTHISI platform [6], to make clear how it facilitates the interconnection and the coordination of a large set of heterogeneous devices, so as to fulfill the aforementioned goals.

[1] http://iot.synaisthisi.iit.demokritos.gr

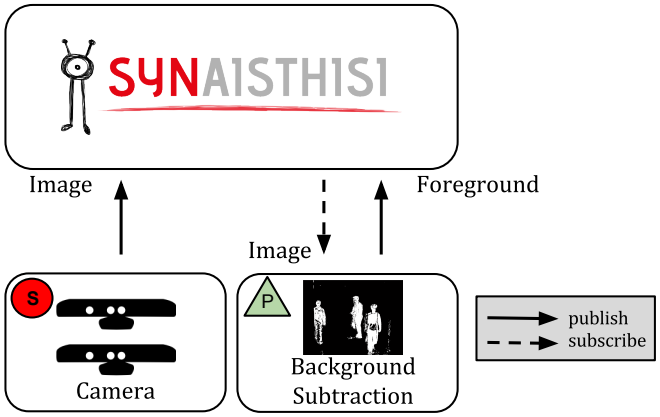


Fig. 1. Two services that are interconnected with the SYNAISTHISI platform and exchange messages.

The SYNAISTHISI platform is composed of five core layers, each providing functionalities at different levels. More specifically, these layers manage the physical objects and their integration with the platform as SPA services, support communication channels to realize the service interconnection, control the available resources, allow custom applications deployment and provide a set of tools as well as visualizers to accommodate system administrators.

In a nutshell, all S-, P- and A-type services communicate by publishing and/or subscribing messages within *topics*, a process managed by a message broker. Particularly, all measurements, decisions and commands are encapsulated into messages and communicated via these topics. Once a message is published on a topic, the message broker informs and delivers it to all clients that are subscribed to that topic. An example depicting this kind of information exchange through topics is shown in Fig.1. Solid arrows correspond to messages published to topics, while dashed arrows depict the messages arriving to the subscribers of those topics. A camera is an S-type service and can be controlled via the platform. It captures RGB-D (i.e. RGB and Depth) video data from the physical world and publishes them to a topic. A Background Subtraction module is a P-type service, which receives the RGB-D data from the camera by subscribing at the same topic. Upon processing, it publishes its output.

### B. Smart Meeting Room Ontology

We next provide an overview of the main concepts involved in the ontology that is utilized by the SYNAISTHISI platform. The Internet of Things - Architecture (IoT-A) ontology [17] is the upper ontology that forms the basis of our semantic model.

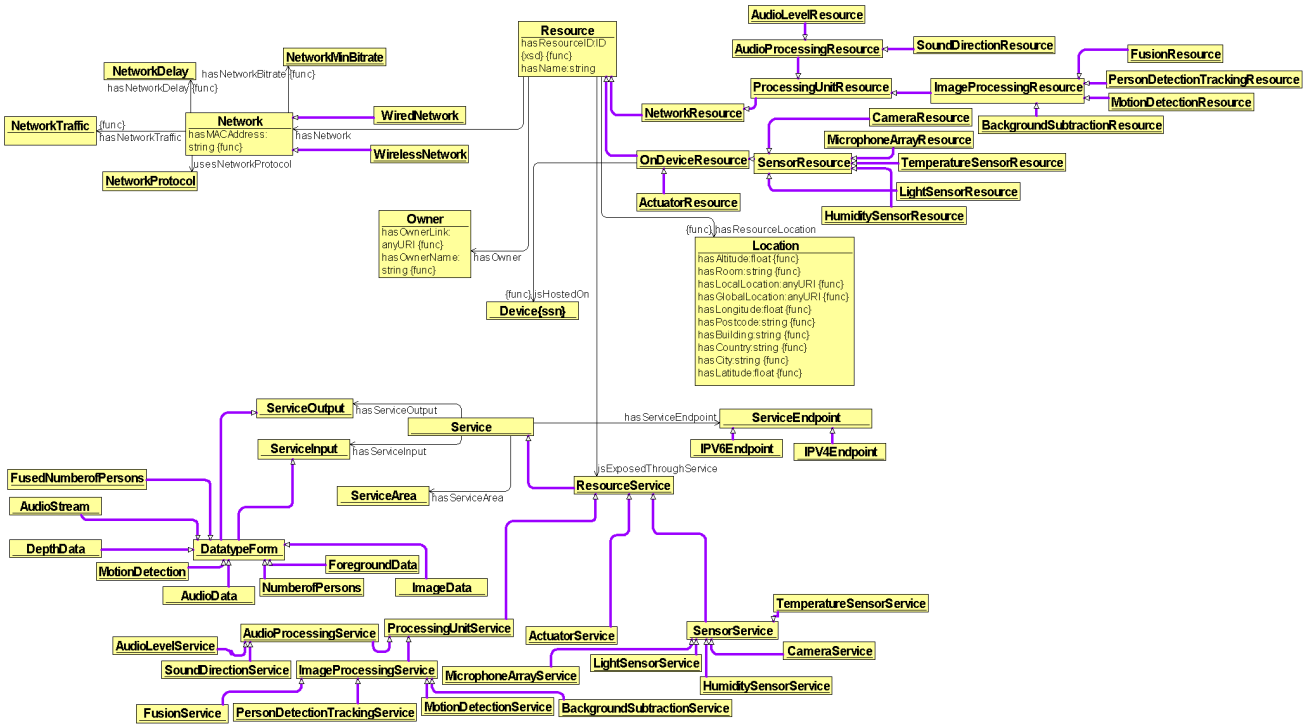


Fig. 2. Part of the core classes and properties contained in the proposed ontology. Thicker arrows depict subclass relations, while thinner arrows correspond to properties.

IoT-A has two core classes, namely Resource and Service, which are of great importance in our case. Resource provides information related to the hardware components (i.e. devices), while Service exposes the functionality of its associated Resource to the outer world as a web service, by describing what the service does (e.g. service input/output, service endpoint, the physical world area affected by the service etc.). These core concepts of IoT-A are extended and specialized in our ontology to represent the services and associated devices of a smart meeting room. Note that the notion of S-, P- and A-type services, in the context of our work, encapsulates both the Resource and the Service concepts of IoT-A.

Specifically, we introduce classes and relations (properties) to represent the various service types. These classes are further refined with subclasses, that describe specific types of *sensors*, e.g. humidity sensors, temperature sensors, cameras and microphones, *processors*, e.g. modules for subtracting the background of a scene, and *actuators*, e.g. plug switches. Other important concepts included in our ontology are the location, the network connection capabilities and the owner of a device, as well as a stack of concepts to capture the various IOPEs (Inputs, Outputs, Preconditions, Effects) involved in smart meeting room services. Moreover, some aspects of sensing and actuating devices are captured using concepts offered by the popular Semantic Sensor Network ontology[2] (SSN). Finally, the qu-rec20 ontology[3] is used to describe physical quantities (e.g. temperature) and their measuring units. A basic fragment of our ontology is depicted in Fig. 2. With the descriptions of the platform and the ontology in place, we now analyze the service discovery and interconnection procedures.

### C. Service Discovery

Service discovery can be facilitated by exploiting the semantic descriptions available in the proposed ontology. A developer can issue through the platform SPARQL[4] queries describing the characteristics of the desired services and recover matching services. An example of service discovery using our ontology is given in Section IV, as part of the "person counting" test case.

### D. Service Interconnection & Complex Applications Delivery

Having discovered the services to use, the next step is to describe their interconnection. This process is referred to by [6] as *SPA piping process*. The piping is achieved by *(i)* assigning a unique identifier to each service and *(ii)* defining which service's output is another one's input. According to [6], sensing services usually receive no input, but produce output that is broadcasted using the platform. On the contrary, actuating services usually produce no output that is communicated through the platform, but can utilize input streams. Finally, processing services, both receive, and produce streams[5].

[2]SSN ontology: http://purl.oclc.org/NET/ssnx/ssn

[3]qu-rec20 ontology: http://purl.oclc.org/NET/ssnx/qu/qu-rec20

[4]SPARQL query language: http://www.w3.org/TR/sparql11-query

[5]In the paper, input and output streams refer to actual information flow. It is possible for sensors to have input and actuators to have output, but only in the form of control signals, which are issued by the platform alone, for example "enable", "disable", "change sampling rate", etc.

A developer has knowledge of how the services should be interconnected, and the platform provides him with the tools to easily describe the interconnection scheme. Initially, the developer must define an identifier for each service (i.e. a service id). Then, the definition of a data flow consists of arranging the service identifiers in a partial order such that the identifier of a service whose output is inputted to another service, precedes the identifier of the latter. As soon as the description is submitted, the platform undertakes to enable the requested services and bring the required communication channels between them into realization. The process that we just described constitutes the submission of an *application blueprint specification* and includes all the necessary steps to realize a complex application.

## IV. Example Use-Case: Smart Room Modules' Interconnection and Complex Application Delivery

In this section we present a test case of the described methodology. In the context of the SYNAISTHISI project, a typical meeting room has been equipped with several sensors, interconnected through the SYNAISTHISI platform, to provide S-type services. Moreover, a number of services performing audiovisual analysis have been developed in order to provide several cloud-based P-type services. These services were semantically annotated using the proposed ontology, and form its instances.

### A. Person Detection and Tracking Module

The Person Detection and Tracking service is a P-type service which aims to continuously detect and track people present in the room. The algorithm applied uses a blob-detection approach, functioning under the assumption that a moving blob at the foreground should belong to a human. When a new person enters the room, he/she is tracked during his/her stay. This service outputs the coordinates of the set of bounding boxes that enclose the detected blobs and also their number. Thus, the number of tracked objects is assumed to be the number of people present within the room.

However, in order for this service to work, several other S- and P-type services should interconnect and exchange information. More specifically, there exist several RGB-D cameras that continuously capture video of the room's interior. Their output is needed from the background subtraction module, which divides video into two parts: the background, which is discarded, and the foreground, which is fed to the tracking module. The latter's output is finally processed by the person detection and tracking module, which publishes the number of people to the platform. The services used are summarized in Table I, while the flow of the information is illustrated in Fig. 3.

### B. Data Fusion Module

Data fusion in general denotes an act of combining data from disparate and even heterogeneous sources, in order to obtain some kind of "improved" information compared to what is possible when each source is used individually. To this goal, the data fusion service aims to combine measurements or raw data from different heterogeneous or homogeneous sensors. In the presented approach, the data fusion is fed with the output of person detection and tracking from two different cameras, i.e., from two different parts of the smart room. Given the locations of the two cameras, results from the two cameras are fused appropriately. In our case, the measurements from two cameras surveying different parts of the room are added to provide the total number of persons in the room.

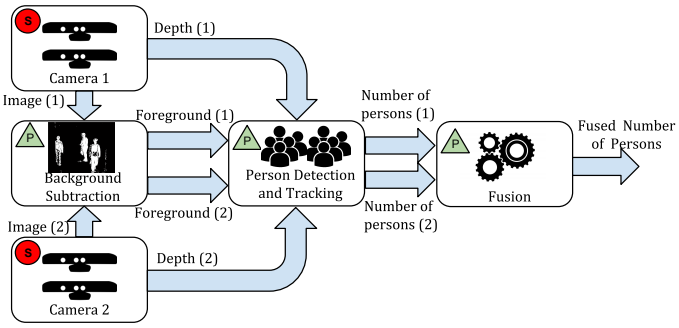


Fig. 3. Flow of information exchanged between S- and P-type services in the person counting module.

### C. Example Service Composition

Let us consider the case of a smart meeting room which is fully covered by two cameras, i.e. "Camera 1" and "Camera 2". Each camera monitors an area of the room, denoted as Area1 and Area2. Also, each camera's output consists of Image Data and Depth Data, which is denoted by "Image (1)", "Depth (1)", "Image (2)" and "Depth (2)" (Fig. 3). A developer that needs to embed a "Person Counting" service to the platform should first discover these cameras. Considering also that this module works on the foreground, he/she also needs to discover a "Background Subtraction" service. Finally, a "Fusion" service whose goal is to combine the individual measurements should also be used.

Assuming the developer has knowledge of the smart meeting room architecture and the kinds of services required to fulfill the given task, he/she is able to discover the necessary sensing and processing services by issuing SPARQL queries to the smart meeting room ontology via the SYNAISTHISI platform. An example of such query to retrieve a camera that observes Area1 of the room (and outputs both Image Data and Depth Data) is shown in Table II[6]. Analogously the developer can locate the rest of the services needed to accomplish his/her task. As a final step, the blueprint specification of the "Person Counting" service must be submitted to the platform (Section III-D).

### D. Re-use of Existing Modules

One of the most important aspects of the SYNAISTHISI platform is the ability to easily reuse all registered services.

[6]The ontology concepts and relations used in this example SPARQL query are depicted in Fig. 2.

TABLE I
S- AND P-TYPE SERVICES USED IN THE PERSON COUNTING MODULE.

| Service | Type | Input(s) | Output |
|---|---|---|---|
| Visual Data | S | - | Image, Depth |
| Background Subtraction | P | Image | Foreground |
| Person Counting | P | Image, Depth | Number of Persons |
| Fusion | P | Number of Persons | Fused Number of Persons |

TABLE II
SPARQL QUERY FOR DISCOVERING A CAMERA THAT OBSERVES AREA1 OF THE SMART MEETING ROOM AND OUTPUTS IMAGE AND DEPTH DATA.

```
SELECT ?cam_res ?cam_serv WHERE {
     ?cam_res rdf:type CameraResource.
     ?cam_res isExposedThroughService ?cam_serv.
     ?cam_serv hasServiceArea Area1.
     ?cam_serv hasServiceOutput ?cam_out.
     ?cam_out rdf:type ImageData.
     ?cam_serv hasServiceOutput ?cam_out_1.
     ?cam_out_1 rdf:type DepthData.
}
```

Since the developer has already described each application in a platform-compatible manner, and has registered it as another service in the platform, then by following the exact same procedure that we have described in the previous sections, it is possible to re-use and re-deploy it. This way, even more complex services can be composed.

## V. CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach for developing complex applications in the context of SYNAISTHISI, an IoT platform. We proposed a smart meeting room ontology and showed how a developer can exploit it in order to discover sensing and processing services, so as to build a complex service, which determines the number of people inside a smart room. The main advantage of our approach, i.e. developing within an IoT ecosystem, is that it does not require any additional expertise by the developer to deliver applications of increased complexity.

Future work shall focus on automatic complex service composition. More specifically, we plan to introduce a mechanism for the automatic discovery and interconnection of services that utilizes semantic information available in the form of ontologies. This way we aim to assist non-developers with experimenting, and quickly developing applications in an IoT ecosystem, using the provided services. Also, we plan to provide customizable user-friendly GUIs, which will aid the developers to easily discover services and more effectively combine them to better suit their needs. In particular, we envisage a GUI where the problem is described in natural language terms and the platform is responsible for the rest (e.g. forming the appropriate SPARQL queries).

## ACKNOWLEDGMENT

This research is part of the "SYNAISTHISI" project results. The project is co-financed by the Greek General Secretariat for R&T, Ministry of Culture, Education & RA and the European RDF of the EC under the Operational Program "Competitiveness and Entrepreneurship" (OPCE II), in the action of Development Grants For Research Institutions (KRIPIS).

## REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] B. Guo, D. Zhang, Z. Wang, Z. Yu, and X. Zhou, "Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things," *Journal of Network and Computer Applications*, vol. 36, no. 6, pp. 1531–1539, 2013.

[4] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, 2011.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols and applications," *IEEE Communications Surveys & Tutoriala*, no. 99, 2015.

[6] G. Pierris, D. Kothris, E. Spyrou, and C. Spyropoulos, "SYNAISTHISI: An enabling platform for the current internet of things ecosystem," in $19^{th}$ *Panhellenic Conference on Informatics (PCI)*. ACM, DOI:10.1145/2801948.2802019, 2015.

[7] D. Guinard, C. Floerkemeier, and S. Sarma, "Cloud computing, REST and mashups to simplify RFID application development and deployment," in *Proceedings of the* $2^{nd}$ *International Workshop on Web of Things (WoT)*. ACM, 2011, pp. 9:1–9:6.

[8] D. Soukaras, P. Patel, H. Song, and S. Chaudhary, "IoTSuite: a ToolSuite for prototyping internet of things applications," in $4^{th}$ *International Workshop on Computing and Networking for Internet of Things (ComNet-IoT), at the International Conference on Distributed Computing and Networking (ICDCN)*, 2015.

[9] P. Stelmach, "Service composition scenarios in the internet of things paradigm," in *Technological Innovation for the Internet of Things*. Springer, 2013, pp. 53–60.

[10] F. A. Kraemer and P. Herrmann, "Creating internet of things applications from building blocks," *ERCIM news*, no. 101, pp. 19–20, April 2015.

[11] S. A. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46–53, 2001.

[12] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic markup for web services," pp. 2007–04, 2004.

[13] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, "Web service modeling ontology," *Applied Ontology*, vol. 1, no. 1, pp. 77–106, 2005.

[14] T. Vitvar, J. Kopecký, J. Viskova, and D. Fensel, "WSMO-Lite annotations for web services," in *European Semantic Web Conference (ESWC)*, 2008, pp. 674–689.

[15] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopecky, and J. Domingue, "iServe: a linked services publishing platform," in *Ontology Repositories and Editors for the Semantic Web Workshop at the Extended Semantic Web Conference (ESWC)*, vol. 596, 2010.

[16] T. G. Stavropoulos, D. Vrakas, D. Vlachava, and N. Bassiliades, "BOnSAI: a smart building ontology for ambient intelligence," in *International Conference on Web Intelligence, Mining and Semantics (WIMS)*, 2012, pp. 30:1–30:12.

[17] S. De, T. Elsaleh, P. M. Barnaghi, and S. Meissner, "An internet of things platform for real-world and digital objects," *Scalable Computing: Practice and Experience*, vol. 13, no. 1, 2012.