IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 28, 2022, accepted April 25, 2022, date of publication May 2, 2022, date of current version May 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3171575

A Survey on the Web of Things

LUCA SCIULLO[1], LORENZO GIGLI[1], FEDERICO MONTORI[1,2], (Member, IEEE), ANGELO TROTTA[1], (Member, IEEE), AND MARCO DI FELICE[1,2], (Member, IEEE)

[1]Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy
[2]Advanced Research Center on Electronic Systems "Ercole De Castro", University of Bologna, 40123 Bologna, Italy

Corresponding author: Luca Sciullo (luca.sciullo@unibo.it)

This work was supported by the EU Electronics Components and Systems for European Leadership (ECSEL) Joint Undertaking (Arrowhead Tools), through the EU Horizon 2020 Research and Innovation Programme, under Agreement 826452.

ABSTRACT The Web of Things (WoT) paradigm was proposed first in the late 2000s, with the idea of leveraging Web standards to interconnect all types of embedded devices. More than ten years later, the fragmentation of the IoT landscape has dramatically increased as a consequence of the exponential growth of connected devices, making interoperability one of the key issues for most IoT deployments. Contextually, many studies have demonstrated the applicability of Web technologies on IoT scenarios, while the joint efforts from the academia and the industry have led to the proposals of standard specifications for developing WoT systems. Through a systematic review of the literature, we provide a detailed illustration of the WoT paradigm for both researchers and newcomers, by reconstructing the temporal evolution of key concepts and the historical trends, providing an in-depth taxonomy of software architectures and enabling technologies of WoT deployments and, finally, discussing the maturity of WoT vertical markets. Moreover, we identify some future research directions that may open the way to further innovation on WoT systems.

INDEX TERMS Deployments, internet of things, software architectures, web of things.

I. INTRODUCTION

Nowadays, the Internet of Things (IoT) can be considered an established paradigm that permeates every aspect of our life. The number of connected devices has by far surpassed the population of the world and it is expected to grow at an exponential pace within the next few years [29]. This makes the IoT one of the hottest topics in research as well as one of the most powerful tools in the industry, with several vertical markets being established, from smart buildings to precision agriculture, just to name a few [4]. At the same time, the continuous placement on the market of new devices and enabling hardware/software technologies, while fueling the success of the paradigm on the one side, has become one of the main causes of fragmentation on the other side, with harmful impacts on most of the existing and the future IoT deployments.

To this purpose, both academic and industrial research highlight the need of an interoperable IoT, intended as the ability to make use of information among heterogeneous providers in a seamless way. Indeed, the lack of interoperability has increased the costs for IoT system deployment/maintenance, due to the need to support heterogeneous interfaces from diverse platforms [106]. At the same time, the possibility to integrate heterogeneous data providers at different scales (devices/platforms/markets) is considered a major source of revenue for next-generation IoT systems [100]. Given the relevance of the topic, many solutions to cope with the fragmentation of the IoT landscape have been proposed in the literature, starting from different perspectives on the interoperability [134], [149]. For instance, Noura *et al.* [106] distinguish among: a (*i*) *Device interoperability*, which is concerned with the exchange of information between heterogeneous devices and communication protocols; a (*ii*) *Network interoperability*, which deals with mechanisms to enable seamless message exchange between systems through different networks; a (*iii*) *Syntactical Interoperability*, rooted on the usage on common data formats such as HTML, XML or ASN; a (*iv*) *Semantic Interoperability*, based on sharing the same meaning for exchanged data; and a (*v*) *Platform Interoperability* which enables IoT applications to integrate data from various IoT platforms. To enable semantic reasoning on the data, it is important to have the available data presented in standard formats, reachable

The associate editor coordinating the review of this manuscript and approving it for publication was Jemal H. Abawajy.

and manageable. This is true not only for the data but also for the relationships among them, that should be made available. This collection of interrelated datasets is referred to as *Linked Data* and it enables a large-scale integration and reasoning on data [61].

Regardless of this categorization, the Web of Things (WoT) denotes a wide range of solutions aimed at achieving interoperability in IoT scenarios thanks to the unifying power of Web technologies. The WoT does not identify a specific product, rather a paradigm complementary to the IoT. It was proposed for the first time in 2008 by Guinard and Trifa [54] and sets the objective of enabling Web-based interactions among IoT devices. This is achieved typically by instantiating a Web counterpart for each IoT device, often called Web Thing (WT), empowering thus a sort of IoT-as-a-Service (IoTaaS). As outlined by Heuer *et al.* [60], "*just as Internet connectivity has enabled intuitive information sharing and interaction through the Web, so the Internet of Things might be the basis for the Web of Things, enabling equally simple interaction among devices, systems, users, and applications*". The advent of the WoT has introduced a major change over classic IoT interoperability approaches [44]. However, because of the lack of standard specifications for developing WoT applications, several reference architectures have been proposed [95], [102], [104]; similarly, there exists a plethora of approaches on how to enable specific features of the WoT, *e.g.* how to support the discovery of WoT devices and how to describe their capabilities in a machine-understandable way [138], [161]. Yet, some de-facto requirements to guide WoT developers and software architects have been provided. Mrissa *et al.* [102], for instance, identified other requirements beside *interoperability*, such as the *live reactiveness* to make the WoT platforms dynamically adapt their behaviors to the environment, and the support for *resource management* to cope with possible connectivity disruptions. Similarly, Guinard and Trifa [55] discussed how to apply the REST principles [39] in WoT deployments composed of resource-constrained IoT devices.

In the industrial domain, the research on IoT interoperability and WoT has led to standardization efforts, which can be considered an effective way to address the fragmentation issue, despite some concerns related to the user acceptance and the temporal effectiveness of ICT standards in general [24]. Beside the initiatives promoted by IETF about REST extensions to model the Things capabilities [57], [84], we highlight the recent release of a WoT standard from the W3C [155], which is intended to enable interoperability across IoT platforms and application domains while preserving and complementing existing IoT standards and solutions. Using their words, the W3C WoT architecture "*is designed to describe what exists rather than to prescribe what to implement*". To this purpose, detailed specifications related to the overall architecture, the interaction patterns and the modeling of WTs have been released [155]; however, they abstract from the way the WTs are implemented internally and from the communication technologies, hence introducing a significant advance compared to the original WoT solutions that were restricted to the usage of the HTTP protocol. Despite the recent appearance, the interest towards the W3C WoT standard is demonstrated by the increasing number of case-studies [3], [22], tools [16], [83] and enhancements [2], [99].

The aim of this paper is to give a *broad* vision of the WoT landscape to both researchers and newcomers, by exploring various facets of such a paradigm, while, at the same time, investigating each of these facets in *depth*. It is worth highlighting that the evolution of the WoT over time is complex and not linear; it encompasses a set of initiatives that took place at the same time and gave different definitions of the main entities. Also, several branches or specializations, such as the Social Web of Things [26] (SocWoT) or the Semantic Web of Things (SWoT) [64], have appeared in the last ten years. From here, there is a need for a systematic review able to frame the main concepts unambiguously while reflecting the appearance of major trends and of new research directions. This paper can be considered both a survey and a tutorial. As a survey, it presents first the historical context of WoT approaches and then reviews the existing WoT architectures, the landscape of the real-world deployments, and the enabling technologies currently available. As a tutorial, it identifies the building blocks common to any WoT deployment, discussing each stage in detail, respectively the WT modeling, WT annotation, WT access, WT discovery, WT mashup, and WT securing. In addition, as a starting point for future research, it points out potentially visionary ideas involving the WoT and its integration with emerging ICT technologies.

The need for a survey on the WoT domain is justified by the growth of the WoT research community, not followed by a consequential increase of summary documents and reference textbooks able to provide a unitary vision of this research area. Indeed, on the one side, we notice the availability of a wide literature related to WoT solutions, with an increasing trend in the last few years favored by the recent standardization initiatives. On the other side, [159] represents – to the best of our knowledge – the only general-purpose survey on the WoT: however, being dated more than ten years ago, it does not cover the most recent evolution of WoT-related concepts and technologies. More recent works [26], [85], [161] focus only on specific, vertical themes (*e.g.* the Thing discovery [161]) but lack the broad vision on the WoT landscape, which instead is one of the main goals of this paper.

In conclusion, the paper is founded on the following key contributions:

1) It gives a global vision on the historical evolution of the WoT concepts, by also categorizing the numerous architectural patterns found in the literature.
2) It provides a taxonomy of the main building blocks which are needed to deploy a WoT system, performing an in-depth literature review for each stage.
3) It outlines the current status in terms of the adoption of the WoT in the real world, reviewing the enabling technologies available in the market and used

by research papers, and discussing the application of WoT approaches to vertical markets of the IoT.
4) It identifies the major open issues and some future directions that may pave the way to next-generation WoT systems.

The rest of the paper is organized as follows: in Section II we introduce the historical background on the WoT, focusing also on the definition of the main entities, Section III presents the architectural paradigms that characterize the WoT, differentiating between earlier works and the W3C standard, Section IV outlines a taxonomy of the main building blocks of the WoT and the current research areas as well as the proposed solutions, Section V introduces the main enabling technologies currently available, Section VI surveys a set of major real-world deployments, Section VII guides the reader towards the open challenges and, finally, Section VIII concludes the paper.

II. BACKGROUND

Although no standard definition exists [142], the IoT is considered as the enabler of intercommunication among heterogeneous devices, while the WoT (WoT) aims at tackling the interoperability issues at the application layer. As highlighted by Negash *et al.* [104], in contrast to the current Internet – also called *Internet of people* [25] –, the IoT, and hence the WoT, enhances the real world by adding connected devices capable of sensing and acting through the Internet. Nevertheless, since the beginning, the WoT has been considered as an extension of the IoT: the main idea is the possibility to enable smart things to communicate through existing and well-established Web standards and Web technologies, thus improving their accessibility and enhancing the possibility of building new applications and services. Continuing improvements both on the electronic side and on the software side led to a new generation of small, cheap, and low power devices. As pointed out by Raggett [118], this is opening the way to a WoT world, where ''Things'' can be considered as proxies for physical or abstract realities and the ''Web'' refers to the possibility that these ''Things'' communicate via Web technologies, such as, but not limited to, the HTTP protocol. In this Section we will give a broad overview of the history of the WoT, the definition of the most important components, and some guidelines that have been looked at by researchers over the last decade.

A. HISTORY/CONTEXT

The concept of WoT started to appear at the beginning of 2000's, when Kindberg *et al.* [79] presented the idea of *Web presence*, an extension of a Web *Home page* to physical entities. A *Web presence* can be considered as a description model of a device (a Thing, as it will be defined later) and the entry point for interacting with it. This is obtained by embedding Web Servers into such Things – *e.g.* printers, projectors, whiteboards, etc. – or by hosting their *Web presence* within a Web Server. In 2007, Wilde [157] proposed to assign a URI to each physical device and to interact with it through the *basic REST verbs*, *i.e.* `PUT`, `GET`, `POST` and `DELETE`. Each device could then be represented as Web resource by using HTML, XML or JSON [18] format. Following the same principle, Guinard and Trifa [54] defined the WoT as a ''*refinement of the IoT by integrating smart things not only into the Internet (network), but into the Web Architecture (application)*''. The same authors in [50] claimed that the WoT needs to extend the existing Web by involving real world objects and embedded devices. Instead of just using Web protocols as means for data delivery – like most of WS-* Web services –, they proposed to make devices an integral part of the WoT by using HTTP as application layer. This can be achieved by making functionalities of the devices available through RESTful APIs over HTTP, and for this reason, the authors identify two possible ways: *Direct integration* or *Indirect integration*. In the first approach devices directly embed a Web server, thus becoming part of the Web, whereas in the second approach an intermediate *Smart Gateway* is in charge of translating REST requests/responses across protocols into device-specific instructions. The latter vision is one of the core concepts that nurtured the FP7 Project COMPOSE,[1] whose aim was to enable an open marketplace of cloud services by combining simple WTs. On a side note, it is intended that COMPOSE is not the only effort aiming to tackle interoperability issues in the world of IoT: we cite explicitly other successful ones like FIWARE [28], SmartSantander, SENSEI [116], [124], Fiesta-IoT [1], BIG-IoT [20], OneM2M [140], Arrowhead [33], OpenIoT [137]. Nevertheless, it constitutes an important historical connotation for our purposes, as it is the first one adopting explicitly the WoT and bringing the concept to the attention of a wider audience [117]. In the subsequent years, the requirements for a system to be considered WoT-compliant had become much less permissive. Some works were entirely dedicated to outline such requirements in a structured way. One of the most representative examples is the contribution by Kamilaris *et al.* [72], in which the requirements of a WoT-enabled Web system were outlined by exploring its components. In fact, in their vision, a system is defined as WoT-compliant when it sticks to a number of precise constraints, for instance (i) the accessibility to WTs at the Web layer through a REST interface, (ii) the discoverability of WTs by means of specific architectural patterns or protocols, (iii) the usage of *multiple* well-known Web data formats and semantics. Other constraints involve security protocols, sharing, physical mashups, and syndication techniques.

Even though small deviations were taken from the original proposal, before the 2010s the research on the WoT proceeded along the same strand, being it not yet a standard, rather an architectural design pattern. However, later on, different currents started to manifest, some to bring in extended concepts, some others to try to establish an actual implementation standard. In the remainder of this section we

[1] https://digital-strategy.ec.europa.eu/en/news/compose-market-place-internet-things

cover the most influential ones. In Figure 1 we also give a pictorial overview of all such currents in the form of a timeline.

1) SOCIAL WEB OF THINGS

The Social Web of Things (normally abbreviated as SWoT, however we refer to it as SocWoT to distinguish it from the Semantic WoT) can be considered as the convergence of Social Web and the WoT, as it enables users to manage, access, share and integrate smart Things/objects through Social Network Sites (SNS) [26]. SNSs are online platforms where people publish, collaborate, and share information with other individuals or groups and build social relationships. Ciortea *et al.* [27] define the SocWoT as the convergence of three dimensions: *pervasiveness*, meaning that Web extends to the physical world by integrating everyday objects and things, *pro-activeness*, since the Web is composed by several entities that, like normal users, produce and consume content by interacting with each other, and, finally, *socialness*, because the Web gathers both human and non-human entities. On top of SoCWoT, a trustworthy Social WoT system is proposed in [65] in which Service Oriented Architecture (SOA) approaches are used to resolve the issue of interoperability. Here, the trustworthiness of the services are calculated by using the friendship and community services offered by the SNSs.

2) SEMANTIC WEB OF THINGS

Technologies for Semantic Web [13] have been playing an important role for exchanging standardized, meaningful and contextualized information in the IoT [120]. We acknowledge several joint efforts (*e.g.* Fiesta-IoT, Big-IoT, OneM2M, etc.). Some of them resulted in standardized information models, such as NGSI-LD,[2] proposed within the context of the FIWARE Project [28] and featuring an API for exchanging contextualized – as in, semantically enriched – information among heterogeneous stakeholders. Semantic Web technologies can also play an important role in the WoT, for which they can be considered as the main interoperability enablers. Researchers formulated several proposals for including the Semantic Web in WoT architecture, the first of which is the so-called *Semantic Web of Things (SWoT)*. As highlighted by Jara *et al.* [64], integrating the Semantic Web into the WoT was seen as the definitive step for reaching what is defined as the *global interoperability*. More in detail, this can be obtained if the information is semantically rich and systems achieve high degrees of autonomic capability of storage, management, and discovery. This way, data is machine-understandable. In [64], authors also outline the fundamental steps for pursuing such a path. A first phase would imply interconnecting Things to the Internet, which corresponds to the growth of the IoT. A second step has the goal to enable seamless interoperability among heterogeneous entities by the adoption of common application protocols, which represents the "basic" WoT. Despite WoT enables technical and syntactical interoperability for diverse kinds of devices, shared data can be represented in different ways, carrying along different meanings and hence hindering the interoperability at the data layer. For this reason, the last step, which is also the main goal of SWoT, is the definition of a common description that allows data to be universally understandable, creating extensible annotations (from minimal to more elaborate ones), ad agreeing on a catalogue of semantic descriptions. This path is actually followed in the recent standardization efforts by the W3C.

[2]https://ngsi-ld-tutorials.readthedocs.io/

3) W3C WEB OF THINGS

In 2014 the World Wide Web Consortium (W3C) manifested interest in the WoT and created the WoT Interest Group, which led to the first official Web Thing Model the subsequent year [53]. The proposal gained consistent momentum and the model has been updated on a constant basis until the latest specification [155] that represents an actual de-facto standard for every component of a WoT ecosystem. For this reason, in this paper we distinguish the W3C WoT, which established a detailed standard at implementation level, from the original proposal, more a set of loose guidelines, that we hereafter label as *Legacy WoT*. This also brings in a consistent change at the architectural level, in fact, as it will be presented in Section III, the concept of layered architecture, which was extensively representative of the Legacy WoT, starts to be replaced by a Thing-centric one. These two approaches are, however, not mutually exclusive, as a matter of fact, one may comply with the W3C standard and yet use it in a new layered architecture [104].

It is worth mentioning that the W3C WoT is *not* the only standardization effort for WoT. An in-depth study on similar proposals was conducted by Martins *et al.* [97] in 2017, where they distinguish among bottom-up approaches, *i.e.* considering each component an atom which can (or cannot) constitute a thing, and top-down approaches, where WTs are seen as "*a cohesive unit of data and functionality*" (more details can be found in Section IV-A). In the first category, they collocate three hypermedia APIs from IETF: JSON HAL [77], HSML [84] and CoRAL [57]. These are mainly syntactical specifications though, in fact they do not include high-level architectural guidelines, semantics, and interaction schemes. Top-down approaches include the Evrythng's Web Thing Model [147] and the Mozilla's Web Thing API [42]. These proposals were considered almost competitors of the specifications proposed by the WoT Interest Group, although not having a similar backing from standardization bodies. However, at the time of writing, they both complement the W3C WoT in a joint effort, while the W3C WoT is emerging nowadays as the most complete and used standard for the WoT.

Table 1 shows an overall comparison between the different WoT versions and evolution stages appeared in the past two decades. In particular, the most important facets of the WoT in Figure 1 are replicated here in the rows, while

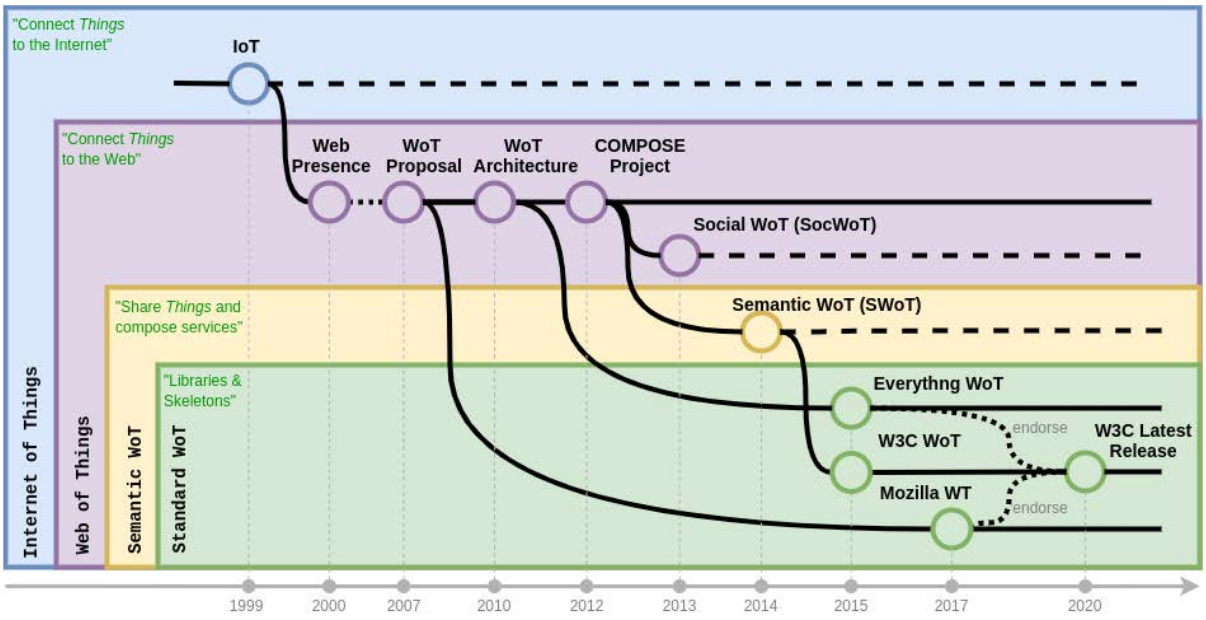


FIGURE 1. Evolution of the web of things over time.

TABLE 1. Comparison of different stages in the history of WoT.

	Enforcing Web technologies	Enforcing SNS interactions	Semantic enrichment	Standardized	Security guidelines	Thing Description	Scripting API
IoT	✗	✗	✗	✗	✗	✗	✗
WoT (2010 Proposal)	✓	✓	✗	✗	✗	✗	✗
SocWoT	✓	✗	✗	✗	✗	✗	✗
SWoT	✓	✗	✓	✗	✗	✗	✗
Everythng WoT	✓	✗	✓[opt]	✓	✓	✗	✗
Mozilla WT API	✓	✗	✓[opt]	✓	✗	✓	✗
W3C WoT (2020)	✓	✗	✓	✓	✓	✓	✓

the columns report a number of selected features that are a discriminant factor between them. Said features are the following:

- If the paradigm considers Web technologies and protocol as a mandatory feature for exchanging data (*e.g.* enforcing the usage of HTTP or CoAP for client-server interactions when possible), then we say that it is "*Enforcing Web technologies*". This is the feature that distinguishes WoT from other paradigms.
- If an application needs to interact somehow with a SNS in order to be considered within the paradigm taken into account, then we say that the latter is "Enforcing SNS interactions". This feature is particularly important for SocWoT.
- If the paradigm considers semantics to be mandatory as part of the data description, then we say that it fulfills "Semantic enrichment". All standardized versions of WoT, plus the SWoT comply with this requirement, which is satisfied by design by using, for instance, JSON-LD.
- If the paradigm is supported by any standardization bodies, then it fulfills the "standardized" feature.
- If a standard specifies security guidelines for connected objects, which may potentially cause physical danger, then it supports the "Security Guidelines" feature.
- If the paradigm enforces exposing and describing things through a standard model, called Thing Description [69], as defined more formally in Section III-B, then it fulfills the "Thing Description" property. This feature is possessed by WoT standards, except the Evrythng WoT.
- Finally, the paradigm may include a "Scripting API", which is a tool for developers that supports the definition of the behavior of the thing. In other words, it guides the development of a Web Thing through standardized APIs and libraries to support the most common operations. This feature distinguished the W3C standard from the others.

B. DEFINITION

In the following, we introduce three key terms that will be used along the paper, specifically the term of *Thing*, of *Web Thing*, and of *Mashup application*. Although it is not possible to provide detailed and universally accepted definitions from the IoT/WoT literature, we aim at identifying the core concepts behind such terms, and at highlighting the differences among them.

1) THING

Generally speaking, in the context of the IoT [46], a Thing denotes a hardware/software entity characterized by

(i) *communication*, (ii) *programmability*, and (iii) *sensing and/or actuating* capabilities. However, such set of minimal requirements is not universally acknowledged. Indeed, according to Szilagyi and Wira [141], an object does not necessarily need native computational and communication capabilities to be defined as a *Thing*, since these features can be acquired later *e.g.* by a chip attached and integrated into the object. Vice versa, a Thing, in order to be defined as such, must possess some interest for services or applications, *i.e.* it must do something useful for an IoT system. Clearly, there exist several kinds of Things at different scales of complexity, like *Tags* (QR Code, RFID), *Devices* (Arduino, Raspberry Pis, etc..), *Machines* (Smart Bulb, smart car), or entire *Environments* (Smart building, smart city).

2) WEB THING

In the following, the Web Thing (WT) is generally referred to as a digital representation of the Thing and of its capabilities via Web technologies. Such a loose definition allows us to introduce a first distinction between the concept of Thing previously introduced, which is anchored to the hardware/-software capabilities of the IoT object, and of the WT, which is an application (software) layer on top of it. However, important differences exist among different WoT approaches on the definition of a WT, and hence on the way to deploy it. For instance, in the Legacy WoT, a WT is mainly a "*digital representation of a physical object accessible via a RESTful Web API*". Hence, the main focus is on access technology (*RESTful Web APIs*). Coming to the W3C WoT, instead, the WT definition comes together tightly with its description model. In particular, the WT is, reporting the working group words, "*an abstraction of a physical or a virtual entity whose metadata and interfaces are described by a WoT Thing Description*." [69]. It is immediately clear how this definition abstracts from the network protocols, rather it focuses on the software architecture guidelines. We detail the differences between the Legacy and W3C WoT approaches in Section III.

3) MASHUP APPLICATION

A *Mashup* application is considered as a software capable of enabling new kinds of services by combining data from multiple sources into a new single output. For the WoT, this translates into an application that consumes data from multiple WTs and arranges it in order to obtain a specific output for scenario-dependant purposes. A typical example is an application that queries multiple WTs associated to thermometer sensors spread over a large environment and simply returns the average. Similarly, Mashup applications can of course involve actuators: in this case, the returning output can be considered as an action to perform. For instance, we assume that our environment includes also WTs associated to smart heat controllers. The logic behind the temperature regulation is implemented into a Mashup application that first retrieves the current temperatures value by probing the WT of thermometers, and then sends to each WT of the controllers the right adjustment. According to Guinard and Trifa [50], [51], *Mashup applications* can be mainly divided into two categories: *physical-virtual mashups*, also called *cyber-physical systems* [92], and *physical-physical mashups*. In the first case, applications are meant to manage data coming from computation and physical processes. In the second case, instead, applications combine real-world services provided only by physical devices. That being said, we recall that Mashup applications are not necessarily WTs, as they might not have any description and not expose any endpoint, however, they necessarily consume endpoints offered by WTs.

C. OTHER INTEROPERABILITY STANDARDS

Over the course of the past two decades there have been major research efforts to address the IoT fragmentation. The WoT in such sense greatly contributes to the body of literature and it is the main focus of the present paper. However, some of the existing solutions reached the stage of standardization in the same way as WoT did through the W3C. Therefore, in order to better frame the W3C WoT within a larger scope, we mention a number of these standards and highlight their main differences. A comprehensive study on these is not the focus of the present paper, as many other relevant studies have performed and in-depth comparison [5], [34], [93] and entire projects were dedicated to this issue (e.g. INTER-IoT [43]). Nevertheless, we believe it is worth mentioning some of the major standardization efforts. OneM2M is a global initiative that brings together major ICT companies and aims to define an interoperability standard, including interfaces, APIs and security guidelines. In particular, it features a common service layer, represented by a middleware, that mediates the communication between use case-specific hardware components and the oneM2M ecosystem [110]. It presents similarities with the W3C WoT, in that both are based on the RESTful design principle, both can bind to various communication protocols and have a way to describe semantic metadata about a thing. However, oneM2M provides a common Service Layer that includes a set of CSFs describing service aspects. By comparison, W3C WoT does not specify common service functions in a TD. Furthermore, a TD can be detached from its WT and act as a complete "user manual", whereas oneM2M does not have such an exhaustive description for interacting with oneM2M devices/resources [158]. NGSI-LD is an ETSI standard that enables actors in a IoT ecosystem to query for contextual information. Group of entities can then be abstracted by an IoT agent which deals with the service layer. It has a similar purpose to the W3C WoT, as it includes semantically-enriched information by design, however, it features a middleware, namely a context broker that mediates every interaction, while W3C WoT allows point-to-point interactions without any central entity [40]. The NGSI-LD data model is adopted by the FIWARE ecosystem [28] to enable communications among IoT agents. The latter are components that handle IoT data heterogeneity and are in charge of translating IoT-specific protocols into the NGSI-LD model. Additionally, IoT Agents map NGSI-LD information as virtual representations of the IoT devices in JSON entities,

stored and managed by Orion, a publish/subscribe context broker. It is important to note that IoT Agent applications are data model-specific, so each different data structure requires a new IoT Agent [163]. Core Specifications by the Open Connectivity Foundation (OCF) are a set of standards that enable IoT interoperability by adding an additional service layer for RESTful client-server interactions [111]. IoTivity is one of the major open source practical implementations of the specifications, which support a way to describe things and organize them hierarchically (differently from W3C WoT, which has a flat architecture by design). However, the OCF specifications is based on CoAP and XMPP only, with a lack of support for other technologies and API patterns.

III. WOT ARCHITECTURES

Beside reconstructing the temporal evolution of core concepts, this survey provides a twofold taxonomy of the existing literature on the WoT, respectively based on (*i*) the building blocks required to set up a WoT scenario (discussed in Section IV), and on (*ii*) the overall architectures that represent the conceptual frames in which the single blocks are deployed. Regarding the latter, we distinguish between two main approaches:

- *Layered* architectures, decomposing the functionalities of a WoT system into separate layers, placed on top of basic M2M communication technologies and IoT protocols. Each layer is independent from the other and can be implemented within a single smart device, within gateways, or spanning multiple network nodes.
- *WT-centric* architectures, envisioning the WT as an atomic set of functionalities and data. The functionalities of a WoT system are built by well-defined interaction patterns among different WTs.

The most prominent example of *Layered* architecture is the one proposed by Guinard *et al.* [52]: from a logical point of view, it extends a network stack (e.g. the TCP/IP stack) with multiple WoT-related application layers on top. Vice versa, the W3C WoT architecture [155] can be considered the most relevant instance of Thing-centric architectures. In the following, we present the two approaches separately.

A. WOT LAYERED ARCHITECTURES

Figure 2 shows the layered architecture proposed by Guinard and Trifa in [55]. The architecture is built on top of IoT solutions enabling smart device connectivity/networking, which are out of the scope of the survey and hence not discussed here. Each layer of the WoT architecture addresses a piece of complexity towards the deployment of highly interoperable, secure, and autonomic WoT systems. More in detail, the *Access Layer* allows turning every Thing into a WT, basically providing each device with REST APIs to support interactions with the external world. The *Find Layer* ensures that the WT can be automatically discovered and consumed while minimizing the need for manual configuration, *e.g.* through the usage of uniform interfaces and data formats describing

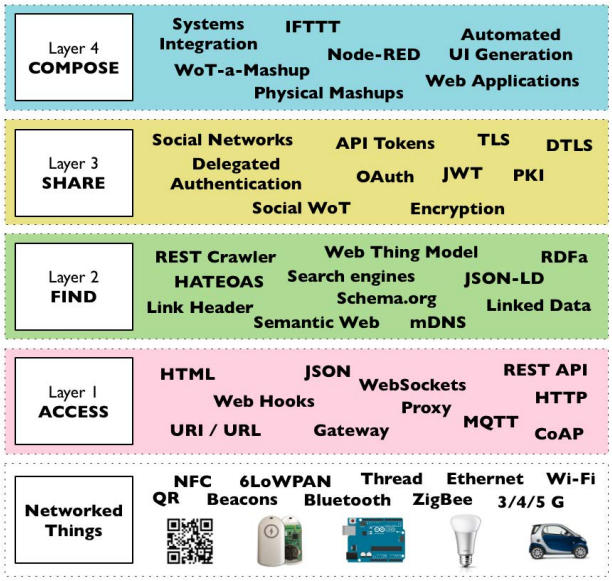


FIGURE 2. Web thing architecture proposed in [55].

the capabilities of a WT. The *Share Layer* defines how data produced by WTs could be shared with external users/applications in a secure and efficient way. Clearly, all the Web security mechanisms (e.g. TLS, OAuth) can be used for this purpose; in addition, SocWoT platforms can maximize the extent of data dissemination [9], [48]. Finally, the *Compose Layer* includes solutions to deploy Mashup WoT applications by wiring together data and services from different WTs. It is worth highlighting that the architecture from Guinard and Trifa [55] is not the only example of layered architecture in the WoT literature [68], although probably it is the most relevant; we cite for instance the four-layer WoT architecture in [95], which includes similar functionalities to those depicted in Figure 2, although at a lower level of abstraction since it is restricted to the usage of the CoAP protocol. More specifically, the proposal of [95] identifies four layers: (i) the *Accessibility layer*, (ii) the *Execution layer*, (iii) the *Proxy layer*, and (iv) the *Composition* layer. The first layer enables smart devices to expose their resources through a common REST interface. On top of it, the *Execution layer* is in charge of virtualizing the physical devices (creating a software clone of it) and of monitoring their connectivity. The third layer enables the communication between the user environment and the running applications and, together with the previous layer, supports the discovery and indexing of the available resources. Finally, the *Composition* layer, like the analogous in [55], provides APIs for mashup application development. There exists layered proposals for the SWoT domain too. In the Web stack for SWoT proposed by Szilagyi and Wira [141], the *semantic layer* is expanded into three sublayers: the *modeling layer*, the *data processing layer*, and the *services and application layer*. In the first layer, semantic web technologies are used to provide a common understanding of

Things' capabilities and characteristics, by employing shared vocabularies and ontologies. The second layer enables reasoning and inference over data by description logic. The last layer supports service publication, discovery, composition, and adaptation.

B. THING-CENTRIC ARCHITECTURES: THE W3C WoT STANDARD

Differently from layered architectures, which manage the complexity of WoT deployments in an incremental way while relying on a loose modeling of the WT structure, Thing-centric approaches envision the WTs as a cohesive unit of data and functionalities [97]. For this reason, they propose well-defined formalisms to describe the WT capabilities (discussed also later in Section IV-A), and, based on that, to characterize the interactions among the WTs. Among the WoT-centric architectures presented in the literature, we illustrate the Web of Virtual Things (WOVT) proposal [104] first, and then we focus on the W3C WoT standard [155], which is becoming the reference solution to deploy WoT systems.

Negash *et al.* [104] proposes the Web of Virtual Things (WOVT) where the core of the architectural design is constituted by the Virtual WT abstraction. In few words, a Virtual WT acts as a proxy for the physical smart device and it is composed of multiple modules to allow the connection between the WT and the physical counterpart, the registration of newly discovered WTs, and the interactions among them. A key component is the WT abstraction, *i.e.* its representation based on three main components: metadata, properties, and actions. The metadata section contains read-only information like the schema models for semantic interoperability. The properties are state variables that can be modified by the physical Thing (*e.g.* sensor values); the actions are methods to alter the behavior of the WT. The fog-cloud architecture permits the Virtual WTs to interact with each other by means of cloud-based gateway that is used also for discovery purposes.

The advent of the W3C consortium in the WoT standardization drew definitely the attention to the definition of what is a WT and how it must be described [155]. Each WT is characterized by the five main modules depicted in Figure 3, and detailed in the following:

- the *Behavior*: it represents the business logic of the WT, *i.e.* the application where both the autonomous behavior of the Thing and the handlers for the WT Affordances (defined below) are implemented.
- *Interaction Affordances*: they represent the interaction model of the WT, specifying how WoT clients can interact with it through abstract operations, hence without referring to a specific network protocol or data encoding. It follows the so-called Properties, Actions, and Events (PAE) paradigm. Each *property* is considered as a state of the Thing and can be retrieved (read) and possibly updated (write). A property can be also *observable*, and in this case the WT is responsible to push the new state to each consumer. An *action* allows a client to invoke a function that typically manipulates a WT state or launches a process. An *event* describes an event source that asynchronously can push data to a consumer.
- *Data Schemas*: they represent the information model (with the related payload structure and data items) to be used in the interaction between the WT and its consumers.
- *Security Configuration*: it contains the security mechanisms provided by the WT in order to control access to its Interaction Affordances. The security configuration includes the *Public Security metadata* and the *Private Security Metadata*. The first component describes the control access mechanisms of the WT, but without including any secret or sensitive data. The second component contains sensitive data to get/obtain access to the WT.
- *Protocol Bindings*: they map the Interaction Affordance to the messages of a specific network protocol (*e.g.* CoAP) and they are serialized as *hypermedia controls*.

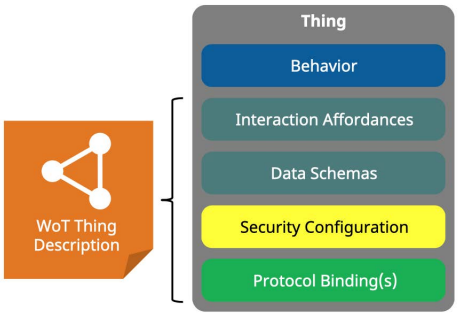


FIGURE 3. W3C web thing architecture proposed in [155].

The first four layers of the W3C WT architecture constitute the Thing Description (TD), *i.e.* a collection of metadata representing the capabilities of the WT in a uniform way. The basic idea is to avoid the definition of yet another protocol to standardize the communications with the WTs, which would increase the fragmentation issue rather than solve it. For this reason, each WT is allowed to support any communication protocol or security mechanism, but it must describe it in TD. As a result, any WoT client parsing the TD and supporting the communication schemes declared in it can instantiate a communication link with the WT. In Section IV-A4, we discuss the WT model of the W3C WoT and provide further details of the TD structure and encoding.

Regarding the deployment scenarios, the W3C WoT standard differs from the architecture in Negash *et al.* [104] which is mainly cloud-oriented. Indeed, the standard is thought to cover almost all the interesting use cases for the IoT and the WoT (Section VI), and hence to be deployed on several network scenarios by means of different interaction patterns. Figure 4 shows all the interaction patterns envisaged by the W3C WoT standard: (i) *device controller*: that involves a local device that is controlled by a user through

a remote controller; (ii) *Thing-to-Thing*: that involves two devices directly communicating with each other; (iii) *remote access*: where a mobile remote controller can switch between different network connections and protocols to meets different security mechanisms; (iv) *smart home gateway*: used for enabling a trusted and safe communication between devices in the local network and external controllers that send commands over the Internet; (v) *edge devices*: that can be considered as augmented home gateways equipped with local computing capabilities to manipulate data; (vi) *cloud digital twin*: that represents a virtual representation of a device or a group of devices that resides on a cloud server or edge device.

IV. WOT BUILDING BLOCKS

Regardless of the architecture in use, the practical deployment of WoT systems requires to address common tasks that can be considered generic building blocks of any WoT system. Indeed, WoT developers must choose proper solutions among a wide set of possibilities to make their WTs accessible, discoverable, and queryable from a network environment, in a secure way. Since the overall goal is to support interoperability, proper formalisms to model and describe the capabilities of a WT could be adopted to enable machine understandability at different extents. Finally, after having addressed the deployment problem of single WTs in isolation, there may be a need to move towards ecosystems of WTs characterized by the seamless wiring of WT data and services. Based on such considerations and on the literature review, we identified six distinct blocks, respectively: WT modeling, WT annotation, WT access, WT discovery, WT Mashup, and WT securing. It is easy to notice that some of these blocks have a correspondence with the layers of the architecture proposed in Figure 2, while others span multiple layers or different components of a Thing-centric architecture. In the following, we discuss each block in isolation; for each, we introduce the scope and we provide a detailed review and classification of the existing WoT studies or W3C WoT functionalities addressing such a task. Figure 5 summarizes the literature review proposed in this Section, with the building blocks and related taxonomy. We summarize in Table 2 the papers cited in this Section, grouping them according to the different building blocks they belong to. For each row we include the reference number of the paper, the specific building block, and a brief description of the contributions.

A. WT MODELING

Through the adoption of Web technologies, WTs are able to exchange data with each other and to be integrated with other components regardless of their implementation; this constitutes the way to support interoperability in many Legacy WoT solutions. However, this does not imply that a WoT client is able to ''understand'' the kind of data or services offered by a WT simply from its URL. For instance, let us consider an IoT-based weather station embedding a Web server and exposing RESTful APIs. A remote controller will then be able to retrieve the current humidity/temperature measurements, or to change the temperature scale by issuing HTTP GET and POST commands. However, the integration still requires some a-priori knowledge, e.g. as how to retrieve the list of HTTP endpoints and which one corresponds to the sensor data access. In the approach proposed in [55] and followed also here, a WT model defines a conceptual schema to describe the resources offered by a WT, such as its endpoints and the input/output data, in a uniform and machine-understandable way. We further distinguish between two sub-components of such schema, *i.e.*: (*i*) the approach used to give a structure to the WoT APIs (called *WT modeling*); and (*ii*) the approach used to add a meaning to such structure by means of Web semantic annotations (called *WT annotation*). Solutions belonging to the first approach are reviewed in the following, while semantic annotations are the main topic of Section IV-B. Clearly, these approaches are often complementary, since semantic annotations can be added to a WT model, like the case of the W3C WoT Thing Description model.

Several WT model proposals exist in the literature, as a result of both standardization processes and research activities. Martins *et al.* [97] provides a qualitative comparison among them; more specifically, they distinguish between bottom-up approaches (*e.g.* the IETF CoRAL [57]), where different units of data are collected inside documents that can represent (or not) individual WTs, and top-down approaches (*e.g.* the W3C WoT) where WTs are envisioned as a cohesive unit of data and functionality like a molecule composed of different atoms. Among others, we focus our review on approaches derived from standardization bodies, specifically the *Media Types for Hypertext Sensor Markup (IETF)*, the *Constrained RESTful Application Language (IETF)*, the *Web Thing Model (EVRYTHNG)*, and the *Thing Description (W3C)*.

1) HSML (IETF)

The Media Types for Hypertext Sensor Markup (HSML) is a standardized data model and interaction model for interoperability promoted by the IETF Thing-to-Thing Research Group [84]. HSML aims to extend the REST paradigm for machine-work flows. To this end, data is serialized in JSON and is structured as a collection of *links* and *items*. Each item (*e.g.* the temperature) can own properties (*e.g.* values and strings) whose keywords are derived from previous IETF standards, such as SenML ([66]), or be associated to a resource type. Also, the specifications allow to describe all the possible CRUD operations on the resource types, and offer link extensions as a way to define custom hypermedia controls.

2) CORAL (IETF)

Constrained RESTful Application Language (CoRAL) [57] is a compact API representation for smart devices with limited capabilities. Similarly to HSML, it is promoted by the IETF and extends the REST paradigm for machine-oriented communications; however, it relies on a binary format

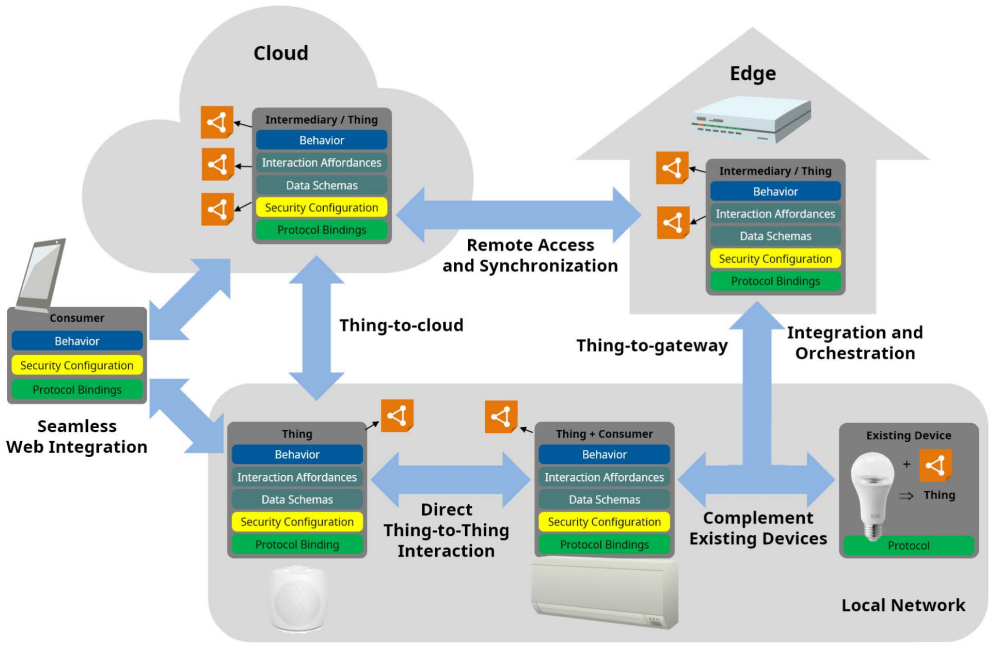


FIGURE 4. W3C web of things architecture [155].

(based on CBOR) to reduce the size of the representation. It is based on the Hypertext Application Language (HAL) and consists primarily of two elements: *links* that describe the relationship between two resources and the type of that relationship; and *forms* that describe a possible operation on a resource as well as the type of such operation. A *CoRAL document* appears as a list of elements, whose types are represented through numbers, together with a hyperlink reference, some options, and an optional body.

3) WEB THING MODEL (EVRYTHNG)

The Web Thing Model has been proposed within the COMPOSE European project, and extensively described in [147]. Part of its concepts has flowed into the W3C WoT specifications described below. The WT model is serialized in JSON and can be semantically enriched. It is composed of four types of resources: (*i*) *properties resources*, that define the *variables* of the WT, *i.e.*, internal states of the WT, (*ii*) *actions resources*, that represent the *functions* offered by the WT that can be invoked by clients, (*iii*) *Things resources*, that contain the list of WTs proxied by this device (used mostly by cloud and gateways), (iv) *model resources*, that contain the metadata providing general information about the WT (*e.g.* its name).

4) TD (W3C)

In order to interact with existing IoT devices and services of silos-based ecosystems, the W3C WoT leverages descriptive metadata, following the architectural style of the Web, *i.e.* REST, with a focus on *hypermedia controls* (*cf.* HATEOAS), and consolidated Web technologies like JSON and Linked Data [14]. The Metadata is serialized into a machine-understandable, but still human-readable, WoT Thing Description (TD [69]), a Web *representation format* based on JSON-LD [76]. One of the key parts of the TD is represented by the concept of *Affordance*. Norman [105] states that *"Affordance refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used"*. Affordances of a WT consist of the information encoded in the high-level interaction endpoints *Properties*, *Actions*, and *Events* (*what* capabilities) and the controls encoded in their forms (*how*), establishing the so-called PAE paradigm.

Listing 1 shows a TD sample for a smart bulb. The bulb has a property called *on* – to represent the current state on/off –, an action called *toggle* – to toggle the bulb –, and the *overheating* event, which is fired whether the bulb temperature reaches a warning level. Since HTML *forms* are controls for arranging the users' inputs on the client-side (based on choices given by the server), the W3C WoT makes them machine-understandable thanks to Web Linking, by providing a context, an operation type, a submission target, a request method, and, optionally, some form fields.

B. WT ANNOTATION

Semantic technologies play an important role in describing meaningfully a WT (and its model), and in enabling autonomic interaction with it. Research about WT annotation can

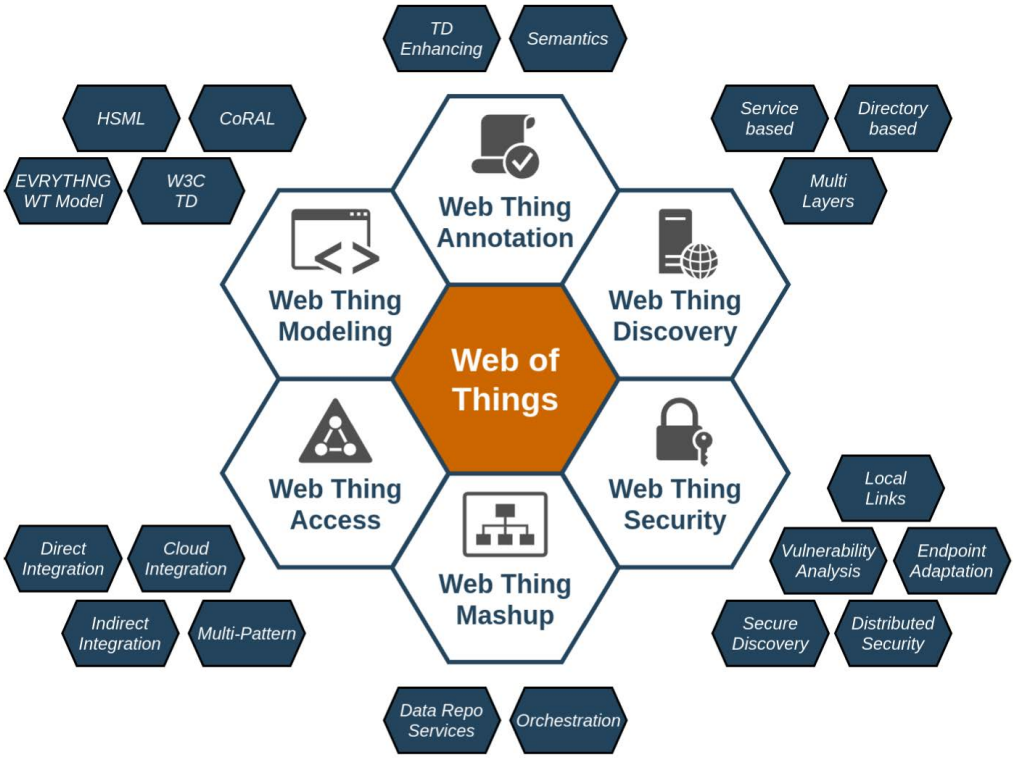


FIGURE 5. Main building blocks of the Web of Things investigated in the literature.

be further divided into (*i*) studies that propose to enrich the WT model with domain-specific ontologies and (*ii*) studies that leverage such annotations to extract knowledge from the WTs. In the first category, for instance, Noura and Gaedke [107] present the Web of Things Description Language for Automatic Composition (WoTDL), an ontology for WoT environments, enabling the automatic selection of devices for sensing and actuation purposes through Artificial Intelligence (AI) techniques. This is just an example of ontologies introduced for the WoT: as pointed out by Li *et al.* [94], users can face difficulties to identify the right ontology for the specific use case, since there exist several standardized – as well as de-facto – ontologies for the WoT, like: the oneM2M Base ontology [110], the SAREF ontology [37], the ETSI ISG CIM Information Model [38], the W3C WoT Thing Description ontology [69] and the W3C SSN and SOSA [152]. In order to provide a better comprehension of the characteristics of each of them, authors of [94] evaluate their coverage on each layer of the WoT architecture discussed in Section III.

Despite the standardization process for the TD [69] is at the conclusive phases, researchers have made several proposals to enhance its descriptiveness on specific application domains. For instance, Sciullo *et al.* [129] introduces a special vocabulary for handling QoS guarantees in the TD, as a bridge towards industrial protocols and deterministic networking. In the same direction, Korkan *et al.* [82] proposes a new vocabulary for TD to describe the sequential behavior of a WT associated with a controller. More in detail, authors introduce the *paths* vocabulary to limit the usage of an interaction, for instance avoiding its execution at any arbitrary time or regulating the multiple interactions flow, hence introducing a sequential order for its execution.

Antoniazzi and Viola [7] present a working implementation of the W3C WoT declined in its semantic flavor: to this aim, they introduce the *SWoT* ontology, which can be used with any standard SPARQL endpoint.

The second category includes studies that leverage the semantic annotation of the WTs to enable WoT services. This is the case of Noura *et al.* [108], which presents an automatic knowledge extraction (KE4WoT) technique to analyze the key topics appearing more frequently in existing ontologies, for a specific IoT application domain. Through the usage of Machine Learning (ML) techniques, KE4WoT is able to identify the most important topics for the *smart home*, *smart city* [96], and *smart weather* domains; based on such output, new vocabularies can be created to support composite applications across diverse WoT ecosystems. Finally, several tools have been proposed for validating the semantic description of resources in IoT/WoT domains, as highlighted by Gyrard *et al.* [56], although only few of them are fully compliant with the target ontologies.

C. WT ACCESS

A key issue in WoT deployments is represented by the access mechanisms that enable the WTs to interact with

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-WoT-SmartBulb",
    "title": "MySB",
    "securityDefinitions": {
        "bearer_sc": {
            "in": "header",
            "scheme": "bearer",
            "format": "jwt",
            "alg": "ES256",
            "authorization": "https://servient.example.com:8443/"
        }
    },
    "security": ["bearer_sc"],
    "properties": {
        "on": {
            "title" : "On/Off",
            "type": "boolean",
            "forms": [{"href": "https://mysb.example.com/on"}]
        },
    },
    "actions": {
        "toggle" : {
            "forms": [{"href": "https://mysb.example.com/toggle"}]
        }
    },
    "events":{
        "overheating":{
            "data": {"type": "boolean"},
            "forms": [{
                "href": "https://mysb.example.com/overheating",
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

Listing 1. Thing description sample for a smart bulb.

other components and hence to operate over the network. This can be achieved with different patterns. More specifically, we distinguish between *Direct Integration*, *Indirect Integration*, *Cloud Integration* or *Multi-pattern Integration* approaches. The first two approaches were originally discussed by Guinard and Trifa [50], while the others are more recent and consequential to the abundance of cloud-oriented IoT platforms and to the proposal of all-embracing solutions like the W3C WoT standard. In the first approach (*Direct Integration*), the smart devices are assumed to be IP-enabled and to own enough computational resources to host a Web server and to expose RESTful APIs, as experienced in [114]. Clearly, this solution poses many challenges on resource-constrained IoT devices, such as the need for adequate computational requirements to host the Web server, although this aspect is mitigated by the recent advances in micro-controllers. Another issue is the support for change-of-state notifications on REST architectures, for which only workarounds can be adopted [78]. Both such challenges can be faced by using the *Indirect Integration* approach, in which an intermediary gateway is employed to connect the smart devices to the WoT. More specifically, smart gateways are in charge of exposing the RESTful APIs on behalf of the smart device, by abstracting its communication endpoints, and of translating the REST requests into open/proprietary communication protocols. This is the case of the WoT system for domestic energy management proposed in [136], where the smart gateway resolves proprietary protocols of heterogeneous devices, in particular translating HTTP requests into the ZigBee commands. In addition, smart gateways can be used to arrange multiple data sources into a higher-level RESTful API, hence serving as orchestrators of services. In [148], the smart gateway is able to manage requests from clients through a RESTful interface and to handle different data types for the WoT resources, mapping each device to the corresponding URI. We highlight that *Indirect Integration* can be effective to extend the WoT towards highly constrained IoT devices like smart tags. For instance, Guinard *et al.* [49] proposes a RESTful architecture for the Electronic Product Code Information Service (EPCIS). Specifically, the goal here is to associate to each tagged object, location, or RFID reader, a unique URL that can be used by Web applications and easily shared across the Internet. This is achieved by means of a *RESTful EPCIS module* that translates the incoming requests into Web Sockets (WS) requests. The third approach (*Cloud Integration*) relies on external platforms to manage and expose the WTs; this introduces other advantages, *i.e.* the possibility to deal with connectivity issues such as Network Address Translation (NAT) and firewall traversals, and to overcome the need for a public IP address for each WT. The Stack4Things middleware proposed by [12] includes open-source tunneled reverse proxy mechanisms capable of exposing Web servers hosted on IoT devices to the public Internet. The system relies on multi-tenant Domain Name System-as-a-Service (DNSaaS) solutions to assign URLs to smart devices as sub-domains of a registered (public) domain, hence allowing to reach the WTs behind NATs. Finally, *Multi-pattern Integration* refers to WoT deployments where multiple approaches (direct/indirect/cloud) are jointly supported to connect the WTs to the Internet; this is the case of the W3C WoT architecture illustrated in Figure 4.

D. WT DISCOVERY

The capability to discover Web-connected embedded devices in real-time and in a scalable and flexible way has been considered the Achilles' heel of WoT by Zhou *et al.* [161]. The absence of a standardized discovery procedure has determined the proposal of many online sensor directory services, starting from the Sense2Web platform [10]. Generally speaking, the findability issue can be split into two sub-problems: (i) how to discover the network addresses of the smart devices hosting the WTs; (ii) how to understand the functionalities offered by the WTs. Regarding the first issue, a straightforward approach is to exploit existing solutions on the Internet, like the DNS: to this aim, the authors of [70] propose the inclusion of a new top-level domain at the DNS to support the discovery of Web-enabled physical devices. The authors of [138] raise the issue of incompatible

islands of WoTs caused by the presence of various Service Discovery Protocols (SDPs); for this purpose, they propose Sherlock-SD, a universal discovery protocol that enables a smart object to discover WoT devices nearby, regardless of the SDP in use. The framework broadcasts probing messages to extract a distinctive attribute from the target WoT device, and hence to identify the SDP. Then, it picks the corresponding SDP adapter or downloads it from a repository. The second issue is more challenging and has opened the way to a plethora of solutions – most of them belonging to the SWoT domain. Based on the software architecture used to support the discovery process, we can group these solutions into three main categories: (i) *multi-layers solutions*, where the discovery process is meant as a combination of procedures that take place at different levels of a layered architecture, (ii) *directory-based solutions*, where the discovery takes advantage of a directory component in order to keep track of the resources of a WoT system, and (iii) the *service-based solutions*, which are implemented as standalone services.

1) MULTI-LAYERS DISCOVERY

Multi-layers discovery solutions are typically based on WoT layered architectures, and the discovery mechanisms are spread among multiple layers. Examples of such mechanisms, derived from the literature on Service Oriented Computing (SOC), include the classical stages of clustering (grouping similar services into clusters), indexing (organizing search key values into catalogs) and ranking (filtering the best services based on the query parameters). An example of discovery architecture addressing all the three functionalities previously mentioned can be found in [103]: here, the authors propose also a global similarity metric which takes into account semantics and type similarity between WTs/services. Ruta *et al.* [123] propose a general framework for the SWoT where the discovery mechanism is jointly provided by: a (*i*) *field layer*, which is in charge to interconnect the micro-devices of the environment with hosts that are able to extract their data; and a (*ii*) *discovery layer*, which supports knowledge dissemination and retrieval. The entire process can be summarized through the following steps: (i) extraction of resource parameters (objects characteristics are shared with the *discovery layer*), (ii) resource information dissemination (diffusion of resource characteristics at the discovery layer), (iii) peer-to-peer collaborative resources discovery, and (iv) extraction of selected resource annotations, that can be used for semantic-based queries and reasoning. Nadim *et al.* [103] propose a distributed three-tier architecture to enable semantic discovery over dynamic WoT environments. In particular, the architecture can handle the dynamicity of WoT services thanks to a clustering algorithm and the mobility of *Wot Gateways*. While the first layer enables the *WT access*, the second layer is responsible of hosting the *WoT gateways*, the discovery functions, the IoT gateways management, and the service composition functions. Each WoT gateway contains the semantic description of the associated IoT service. Finally, the third layer contains discovery and indexing servers, whose goal is to issue the query to be submitted to the WoT gateways, manage a geospatial index to discover the most suitable WoT gateways based on geographical features, and to rank/visualize the results.

2) DIRECTORY-BASED DISCOVERY

Directory-based discovery solutions exploit the capabilities of a *directory* component, which is mainly characterized by two main functionalities: (i) the *registry*, *i.e.*, the possibility to store and retrieve resources through data queries, and (ii) a *subscribe* mechanism that allows clients to be notified as soon as new resources are registered to the directory. Regarding the first functionality, we cite the platform in [89] which consists on a semantic web-enabled repository providing platform independent APIs based upon semantically enriched metadata. The platform supports RDF-based IoT data sharing and retrieval across heterogeneous platforms as well as the registration of Uniform Resource Identifiers (URIs) as entry points of the WTs acting as proxies for physical and abstract entities. The IoT registry has been tested in large-scale environments with thousands of devices belonging to different testbeds realized within the FIESTA-IoT project [1]. In the architecture proposed by Mathew *et al.* [98], the knowledge base acts also as a directory service for ubiquitous context-aware applications. A knowledge base server is used to register all the WTs and to maintain their profiles. In [7], the authors present a general ontology to describe the WT profiles and to enable the WoT discovery on top of the SEPA, a SPARQL event processing architecture used as SPARQL endpoint [32], [121]: in addition, the SEPA broker acts as a WT directory and notifies any semantic status change of the directory through a publish-subscribe mechanism. The SEPA notification mechanism is used by the framework described in [128] [130]; here, users can register to the discovery service through a semantic query, and be notified as soon as new WTs that match the semantic descriptions become available within the WoT system.

Since the W3C WoT Discovery is still under standardization process [156], several works proposed solutions to ease the search operations of the TDs. Leveraging the Eclipse Arrowhead architecture [8], Sciullo *et al.* [131] propose to use the Arrowhead *Registry* component as directory for the WoT services. Through the *WoT Arrowhead Enabler (WAE)*, each WT is mapped and registered as an Arrowhead service, hence becoming discoverable and usable by Mashup applications. Another example of integration between discovery services and the W3C WoT ecosystem is given by Novo and Francesco [109], where authors design an architecture for enabling the discovery using both a (WoT) Thing directory and the Resource Directory according to the IETF Core Resource specifications. More in detail, they extend the *Resource Directory* with additional semantic information to enable an interconnection with the Thing directory, granting full compatibility with the W3C WoT standards.

3) SERVICE-BASED DISCOVERY

In the Service-based discovery solutions, the discovery process is typically granted by ad-hoc software processes that can be considered as *standalone* services. This is the case for instance of García Mangas and Suárez Alonso [44], where authors propose a *Python framework* for the W3C WoT and present a *discovery module* based on DNS Service Discovery (DNS-SD). The latter defines how to use DNS to browse and register services in a network, hence providing a decentralized service discovery solution. The resource discovery process presented in [95] exploits the border routers of the architecture, *i.e.* those components that directly communicate with the WTs and are aware of their IP addresses. The discovery is implemented as a process that simply queries the border routers for retrieving the IP addresses and stores the reference of the WTs in a resource directory. Semantic search engines can be used to discover linked data endpoints of the WTs: this is the case of the WoT Semantic Search Engine (WOTS2E) [72], which proposes to use SPARQL queries to verify the validity of the endpoints, and the presence of ontologies relevant for the IoT/WoT context.

E. WT MASHUP

WoT Mashup applications are enabled by the capability to gather and manage data from multiple, potentially heterogeneous WTs in a seamless way. In the following, we distinguish between two categories of applications: (*i*) repository services, acting as collectors of data from the WoT scenario and enabling data sharing and aggregation from multiple WTs and Web sources, and (*ii*) composition toolkits, enabling automatic WTs orchestration. The first category includes many studies from the SocWoT literature, in which the SNSs are used to share WoT data thanks to their embedded trust mechanisms (*e.g.* based on the notion of friendship); however, we exclude here traditional IoT platforms offering Web dashboard for sensor data visualization or management, like ThingSpeak[3], ThingsBoard,[4] Azure IoT,[5] etc. For the second category, we focus on solutions enabled by the usage of uniform and well-defined WT models.

1) DATA REPOSITORY SERVICES

SNS platforms have been used on top of WoT deployments for two main purposes: (*i*) as authentication authorities and/or (*ii*) as shared data repository. As an example of the first purpose, Shoaib *et al.* [135] propose a WoT system where the access to the sensor data is controlled by SNS open API authentication mechanisms. The second category is quite popular although it implies that all the IoT devices are connected to the Internet and natively support the APIs offered by the SNS. This is the case of the study proposed by Baqer [9], where each IoT device publishes its own sensor data on a dedicated social page. *Inter-portal communication* facilities of the SNS, like associating followers and friends in Twitter and Facebook, allow users/applications to access the sensor data published on other accounts. For instance, the SenseShare application [127] proposes to use *Facebook* as main frontend, by taking advantage of the authentication, privacy, and security functionalities offered by the SNS to support the data sharing. Other studies propose the usage of an authentication proxy between clients (*e.g.* Web browsers) and the WTs, in order to support several SNS through a single trusted connection. This is the case of the Social Access Controller (SAC) proposed by Guinard *et al.* [48], which acts both as an authentication layer and as a controller: its main role is to retrieve data from the APIs of the smart device and to publish it on different SNSs. A similar approach is also considered by Kamilaris and Pitsillides [71] to share smart home data across SNSs: the Web-enabled devices communicate directly or through a smart gateway with a proxy server that sends data to SNSs through their REST APIs. In [160], authors propose a framework enabling social interactions between humans, humans and smart devices, and smart devices only. The platform stores all the received data from the IoT, by adding semantic annotations to the collected data; an NLP component is in charge of translating the raw data to natural language sentences which can be published on the SNSs. Paraimpu [115] is a software platform allowing the end-users to connect, use, share and compose physical and virtual Things, including not only hardware devices, but also existing Web services and SNSs. As a result, users can interact with WTs through the SNS, sending the commands to be executed on specific actuators. Finally, we register a considerable interest in the bridging of Body Sensor Networks (BSNs) into the WoT via SNS platforms. Among others, we cite the pioneering study in [35] and the SenseFace [119] platform. In the first paper, the authors propose to integrate BSNs and SNSs so that only the authorized users can monitor in real-time the data coming from other members of the BSN. In the second paper, the authors present a framework to gather heterogeneous sensory data from a BSN and to publish them to a SNS in real-time, by coping with the devices' mobility.

2) WT ORCHESTRATION

The usage of RESTful interfaces and WT models provides a straightforward solution to face the heterogeneity of smart devices, which in [23] is mentioned as the main obstacle toward the creation of physical mashup applications for the IoT. One of the first examples of a WoT Mashup application based on RESTful APIs is the WoTKit framework [15] which introduces capabilities of data aggregation, visualization, and processing. The programming paradigm is based on processing pipes built by users to generate new sensor data from Web sources. In Ventura *et al.* [150], authors propose to describe WoT APIs through the RESTdesc [151] specifications, and to use JSON-LD as data exchange format. Then, they demonstrate how to deploy Mashup applications that generate plans by using standard semantic Web reasoners. The arrival of the W3C WoT enforced such research

[3]https://thingspeak.com
[4]https://thingsboard.io
[5]https://azure.microsoft.com/overview/iot

direction since it provided a uniform way to describe the WT interfaces. Despite the freshness of the standard, we register a considerable number of studies proposing mashup toolkits for W3C WT scenarios. Among others, we review the proposals in [75], [83], [91] and [130]. In [75], the authors propose to extend the TD concept to describe WoT Mashup applications; the representation includes objects to express functionalities and data that are externally accessible via WoT Interaction Affordances. In addition, the authors show the equivalence of the TD representation and UML Sequence Diagrams, and hence the possibility to switch from a graphical representation (human-understandable) to a standardized textual way (machine-understandable). Finally, they describe a tool for the automatic conversion and code generation. The same approach is extended in [83] for the case of automatic generation of Atomic Mashups (AMs). In [91], the authors point out the need of a descriptive language to model the capabilities of a hierarchical group of WTs always interacting with each other; such WT group is called an Asset in [91]. Hence, they propose to extend the W3C WoT TD to describe the resources of an Asset; at the same time, they consider a graphical interface (Asset Composer) to manage and create Mashup applications through drag and drop actions. The WoT Store [130] is a framework for the easy management of WTs and applications adhering to the W3C WoT standard. Specifically, it allows users to control their WTs throw a Web dashboard; the most interesting feature is that the Web rendering is done automatically, based on the PAE model of the WT. In addition, the tool works as a repository of WoT Mashup applications, *i.e.* users can search the software that matches the characteristics of the existing WTs through SPARQL queries [128]. Finally, some studies focus on mining the collaboration relations between WTs, in order to incorporate them as recommendation for mashup creation. This is the case of the work in [143], in which the authors propose to retrieve the metadata of Web APIs and the historical collaboration information in order to mine the association rules among tags and to derive the Web API collaboration patterns.

F. WT SECURING

Security is a key requirement for most of WoT systems, and it is also a wide area including issues of authorization to WT functionalities, protection of the underlying Thing, WT data integrity, and confidentiality, just to name a few. At the same time, we remark that most of the WoT security challenges recall the same challenges that affect the IoT and the Web in general, and for which many solutions have been proposed so far [45] [87], [126]. This holds in particular for the W3C WoT, since, as highlighted in [154], "*The Web of Things is descriptive, not prescriptive, and so is generally designed to support the security models and mechanisms of the systems it describes, not introduce new ones*". Nevertheless, an explicit effort has been made by the W3C WoT working group to identify and provide some guidelines concerning security and privacy issues [154]. Not surprisingly, the major concerns occur during the *security provisioning* phase, i.e. when setting the security metadata and credentials needed during the operational state of the WT, and during the *maintenance and update* phase. Also, a *WoT Threat model* has been proposed with the aim of reviewing the security stakeholders, the security-relevant assets, the possible attackers, and the attack surfaces of a WoT system [154]. Still, the descriptive approach can introduce some vulnerabilities, as pointed out by McCool and Reshetova [99]: some of them are specific of the W3C WoT standard, others can affect the WoT in general. We describe some of them in the following:

- *Local Links*: secure application protocols widely used in the Web, like HTTPS, might not work for local links because of a possible lack of connection to the Internet (needed for instance for the certificate revocation check). Hence, the TD needs to take into account how to manage the description of such local links.
- *Vulnerability Analysis*: while the TD is meant to ease the discovery of resources, it may make life easier for attackers, who can discover vulnerabilities and infer private information about a WT simply by accessing such description.
- *Secure Discovery*: semantic discovery implies semantic query operations, which are often using significant amounts of network and computational resources, hence again offering the opportunity for DDoS attacks.
- *Enabling Distributed Security*: a message from one WT to another may travel across several different intermediate entities, like proxies or gateways. The security metadata included in the TD should enable end-to-end communication, while at the same time it must include all the information required for granting authorization and authentication at each stage.

V. ENABLING TECHNOLOGIES

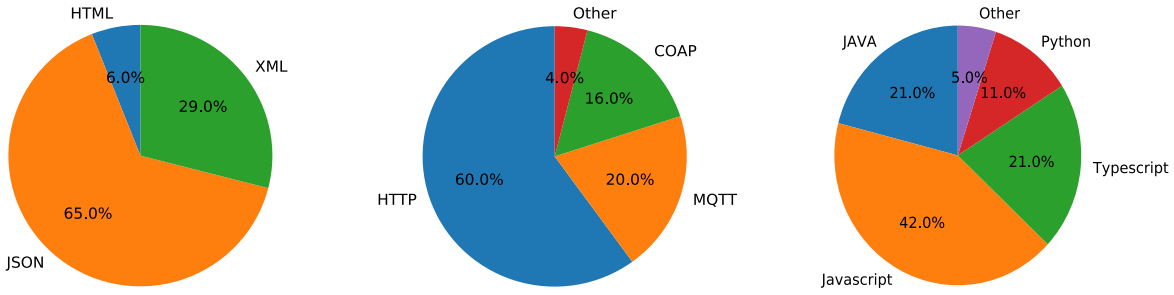
Instantiating a WT from scratch is often a complex task and involves the usage of multiple technologies. Further instruments and expertise may be required to set up a WoT scenario composed of multiple, interacting WTs and Mashup applications as data orchestrators. This is why the WoT research community contributed over the years to the realization of a Software ECOsystem (SECO) composed of tools whose goal is to ease as much as possible the adoption of the WoT paradigm on top of an IoT system. In this Section, we review the main enabling technologies of WoT deployments, by distinguishing between: (*i*) basic enablers, constituted by data formats, protocols, and programming languages (Section V-A), and (*ii*) tools of the WoT SECO, further classified based on the phase of deployment in which they are used (Section V-B), and referring almost exclusively to the W3C WoT.

A. DATA FORMATS, WEB PROTOCOLS, PROGRAMMING LANGUAGES

We propose a quantitative analysis to assess the popularity of the main data formats, Web protocols, and programming languages used in WoT deployments. More in detail, we consider

TABLE 2. List of investigated research works with respect to the WoT building block they refer to.

Reference	Block Name	Description
[55]	Modeling	Definition of a conceptual schema to describe the resources offered by a WT
[97]	Modeling	A qualitative comparison between different WT model proposals
[84]	Modeling	Standardized data model and interaction model for interoperability
[57]	Modeling	Compact API representation for smart devices with limited capabilities
[148]	Modeling	Detailed description of the Web Thing Model
[105]	Modeling	A guide on how to design intuitive interfaces for common things
[106]	Annotation	Web of Things Description Language for Automatic Composition
[94]	Annotation	Evaluation of different ontologies inside WoT architecture layers
[130]	Annotation	Definition of a special vocabulary for handling QoS guarantees in the TD
[82]	Annotation	Vocabulary for TD to describe the sequential behavior of a WT associated to a controller
[7]	Annotation, Discovery	SWoT ontology usable with any standard SPARQL endpoint
[108]	Annotation	Technique to analyze the key topics appearing more frequently in existing ontologies
[96]	Annotation	Description of the concept of smart city ecosystems
[56]	Annotation	Tools for validating the semantic description of resources in IoT/WoT domains
[48]	Access	Discussion of different mechanisms for WTs interaction with other components
[114]	Access	A study on the benefits of Web Technologies for Prototyping the Internet of Things
[78]	Access	Workaround to support change-of-state notifications on REST architectures
[136]	Access	WoT system for domestic energy management with resolution of proprietary protocols of heterogeneous devices
[147]	Access	A gateway architecture that enables to access sensor nodes through a RESTful interface
[51]	Access	RESTful architecture for the Electronic Product Code Information Service (EPCIS)
[12]	Access	A middleware capable of exposing Web servers hosted on IoT devices to the public Internet
[161]	Discovery	A survey on the state-of-the-art of search methods for the Web of Things
[10]	Discovery	A linked-data platform to publish sensor data and link them to existing resource on the semantic Web
[71]	Discovery	A top-level domain at the DNS to support the discovery of Web-enabled physical devices
[138]	Discovery	A universal discovery protocol that enables a smart object to discover WoT devices nearby
[103]	Discovery	A distributed WoT services semantic discovery architecture with clustering, indexing and ranking mechanisms
[123]	Discovery	A general framework for the Semantic Web of Things based on an evolution of classic Knowledge Base models
[98]	Discovery	A semantic Web-based architecture to integrate things as Web resources
[121]	Discovery	A SPARQL event processing architecture on top a SPARQL endpoint
[32]	Discovery	A OSGi semantic information broker implementation for IoT interoperability
[128]	Discovery, Mashup	A novel software platform supporting the distribution, discovery and installation of applications for the W3C WoT
[131]	Discovery, Mashup	A platform to manage applications for W3C WoT that exposes both a web and a Node-RED-based interface
[132]	Discovery	A proposal to use the Arrowhead Registry component as a directory for WoT services
[109]	Discovery	A solution that extends the Web of Things architecture to achieve a higher level of semantic interoperability for the IoT
[44]	Discovery	An experimental framework based on the W3C WoT that includes a set of protocol binding implementations
[95]	Discovery	A software architecture to easily mash-up constrained application protocol (CoAP) resources
[72]	Discovery	A comprehensive list of WoT elements based on an analysis of 26 platforms and frameworks
[90]	Discovery, Mashup	Semantic IoT registry of IoT data and WT proxies, proposed and validated within the Fiesta-IoT project
[135]	Mashup	A WoT system where the access to the sensor data is controlled by SNS open API authentication mechanisms
[9]	Mashup	A framework for employing social networks to facilitate the collaboration and coordination of distributed sensor networks
[127]	Mashup	An application that exploits social networks to provide privacy and security while sharing sensor data with other people
[50]	Mashup	A platform that enables people to share their Web enabled devices so that others can use them
[70]	Mashup	Social networking infrastructures and their Web-based APIs in order to integrate Smart Homes to the Web
[160]	Mashup	A Social Web of Thing Framework based on the RESTful Web Service and Social Network
[115]	Mashup	Prototype of a scalable architecture for a large scale social Web of Things for smart objects and services
[35]	Mashup	An architecture for the integration of body sensor networks and social networks through the IP Multimedia Subsystem
[119]	Mashup	A framework to gather heterogeneous sensory data from a BSN and to publish them to a SNS in real-time
[23]	Mashup	A lightweight IoT service mashup middleware based on REST-style architecture for IoT applications
[15]	Mashup	A IoT mashup toolkit that introduces capabilities of data aggregation, visualization, and processing
[150]	Mashup	A proposal to describe WoT APIs through the RESTdesc specifications, and to use JSON-LD as data exchange format
[151]	Mashup	A specification that expresses the semantics of Web services by pre and post conditions in simple N3 rules
[75]	Mashup	An extention of the TD useful to describe WoT Mashup applications
[83]	Mashup	A method that takes the TDs as input and uses different techniques to automatically explore and reduce the design space
[91]	Mashup	A descriptive language to model the capabilities of a hierarchical group of WTs
[143]	Mashup	An approach to mining collaboration patterns between APIs to aid mashup creation for the WoT
[87]	Securing	A report on the various types of attacks and vulnerabilities in the various layers of the WoT architecture
[126]	Securing	Security challenges related to WoT such as unauthorized access, eavesdropping, DoS attack, tempering, and impersonating
[99]	Securing	Analysis on possible vulnerabilities introduced by WoT Thing Description metadata



(a) (b) (c)

FIGURE 6. The percentage of data formats (Figure 6(a)), protocols (Figure 6(b)), and programming languages 6(c)) used by the WoT deployments reviewed in Section VI.

TABLE 3. Comparison of different technologies used by the WoT deployments reviewed in Section VI.

Study	Data Format	Web protocols (Access approach)	Programming Languages
Ruppen et al. [122]	-	HTTP/HTTPS, WS	-
Corredor et al. [31]	CAP (XML), JSON	HTTP (Webhook), HTTPS	Android, JAVA
Paganelli et al. [112]	HTML, XML, JSON	HTTP	JAVA, Javascript
Karim et al. [73]	JSON	HTTP	C
Chakraborty and Datta [22]	multiple (W3C WoT)	multiple (W3C WoT)	Android/NodeJS
El Jaouhari et al. [36]	-	WEBRTC/HTTP	NodeJS, Javascript
Thuluva et al. [145]	multiple (W3C WoT)	-	-
Patel et al. [113]	-	-	-
Klotz et al. [80]	-	COAP, HTTP	JAVA
Ji et al. [67]	-	HTTP	-
Tobarra et al. [146]	-	MQTT	NodeJS
Sciullo et al. [129]	JSON	MQTT, HTTP, COAP	NodeJS, Typescript
Terius-Padron et al. [144]	JSON	HTTPS	-
Bauer et al. [11]	-	-	-
Sciullo et al. [130]	XML, JSON	MQTT, HTTP, COAP	NodeJS, Typescript
Collins et al. [30]	JSON	HTTP	Python
Aguzzi et al. [3]	JSON	HTTP	NodeJS, Typescript
Ibaseta et al. [62]	-	HTTP	Python
Zyrianoff et al. [163]	JSON	HTTP, MQTT	NodeJS, Typescript

the technologies adopted in the WoT use-cases discussed in Section VI, by restricting our study to those papers describing real-world implementations of WoT systems. As shown in Figure 6(a), three data formats have been taken into account in those deployments, respectively the *JSON* (65%), the *HTML* (6%), and *XML* and related dialects, like *CAP* (29%). The result is not surprising, since *JSON* is probably the reference format for data exchange on the Web, and it perfectly matches the WoT requirements. Figure 6(b) analyzes the protocols used to access the WTs (see Section IV-C). Clearly, *HTTP* is the main solution adopted (60%), but it is not the only one. The *MQTT* protocol (20%) is mainly used for push-based communications while *CoAP* (16%) is useful in scenarios characterized by the presence of constrained devices. The *Other* voice includes, among others, the *Web Sockets (WS)*, whose usage is particularly popular in the Legacy WoT for handling bidirectional communications between devices [55]. The statistics about usage of programming languages shown in Figure 6(c) takes into account only the pieces of software that are involved in at least one of the *Building blocks* presented in Section IV, without distinctions whether they are used for the backend or the frontend implementation. *Javascript*-based frameworks represent the 42% of the total cases. This includes also *NodeJS*, mainly adopted for WoT server-side applications, while *Typescript* covers the 21% of deployments, thanks to its adoption by the *Node-wot* [153] – the official W3C WoT implementation framework. The rest is distributed among *JAVA* (21%), which includes also native *Android* applications, and few *Python*-based (16%) and *C*-based deployments (4%). Table 3 shows what data format, web protocols, and programming languages have been used in each study considered for this quantitative analysis. Each row contains the details about a single study, or an empty cell in case no explicit detail is provided in the paper about that specific set of technologies.

B. TOOLS

We group the WoT tools into three macro categories: (*i*) the *Design-Time Tools*, (*ii*) the *Runtime tools*, and (*iii*) the *Management tools*. The distinction follows a *functional* characterization, based at which stage of deployment of a WoT system they are used (Figure 7). In particular, the *Design tools* are used to model the capabilities of WTs and Mashup applications, the *Runtime tools* support the code execution, and the

Management tools are meant to orchestrate and manage the resources of a WoT scenario.

1) DESIGN-TIME TOOLS

A Design-Time tool aims at easing the modeling and the coding process of WTs. Several works focused on the recent W3C WoT, in fact tackling the acerbity of the standard through the proposals of new tools that automatize the code generation and support the sharing of the TDs. This is the case of [74], where authors propose a format to represent Mashup application called *System Description* (SD), an enhancement of the W3C WoT TD, and provide also a tool to generate executable code that implements the business logic expressed by the SD. Similarly, Novo and Francesco [109] creates a visual tool user through which developers can manually translate the TDs for those IoT devices that are not able to parse the TDs. About TD sharing, WoTify [81] is a tool for sharing and re-using TDs and Thing Application made available by the users: the ambitious goal is to create a community that actively contributes to the diffusion of the W3C WoT by easing the mapping of new and existing IoT devices.

2) RUNTIME TOOLS

These tools allow to instantiate a WT, *i.e.*, to turn a WT into a software object, indeed dealing with the *runtime* phase. The software stack implementation of the W3C WT shown in Figure 3 is called a *Servient*: as the word itself suggests, it can behave both as a server and a client.

In brief, the Servient is in charge of parsing and generating TDs, and of supporting multiple Protocol Bindings to enable interactions with different platforms and devices. Using the W3C WoT terminology, a WT is implemented by a Servient, which exposes a representation of it called *Exposed Thing* and makes available to *Consumers* the network-facing interfaces of a WT. Similarly, *Consumers* are software clients enabled by a Servient which again is in charge of parsing the TD and of enabling the proper network stack according to the WT capabilities. Once the WT has been handled by the Consumer, the Servient generates the so-called *Consumed Thing*, a software object allowing to access its properties or to invoke its actions while hiding all the details of the remote communication.

Figure 8 depicts all the modules composing a Servient:

- The *Behaviour* defines the application logic of a Thing, and includes the *autonomous behavior* (the internal functioning of sensors and actuators), the *handlers* of the Affordances (which operation to perform when an Affordance is activated), the *consumer behavior* (controlling Things or running mashups), and the *intermediary behaviour* (proxying or aggregating Things).
- The *WoT Runtime* represents the implementation of the interaction model and the execution environment for the WT Behaviour, being able to fetch, parse, serialize, and serve TDs. The optional *Scripting API* component defines the application-facing interface that follows the Thing abstraction, enabling the Behaviour to run through the application scripts. The WoT Runtime is in charge of instantiating the software representations of the WT, *i.e.* the Exposed and Consumed Things previously mentioned. The *Private Security Data* is managed by the WoT Runtime, but intentionally kept apart from the application and used by the Protocol Bindings to authorize and protect the data integrity and confidentiality.
- The *Protocol Stack Implementation* implements the network-facing interfaces of an Exposed Thing, allowing the *Consumers* to access the WoT Interfaces of a remote WT via its Consumed object. More in detail, the Protocol Stack generates the network messages according to the selected protocol (*e.g.* CoAP). Clearly, multiple protocols can be supported at the same time.
- The *System API* aims at supporting the implementation of the Behavior in case of Things connected via proprietary protocols or protocols that are not natively WoT-enabled.

The *Servient* architecture is agnostic of the programming language used for its implementation. At present, *Node-wot* [153] is the official open-source implementation written in *Typescript* and maintained by the W3C working group. Besides it, within the SANE project [17], researchers released a JAVA version [125] of the *Servient*, even if they claim to support only the subset of specifications of interest for the project. García Mangas and Suárez Alonso [44] propose a Python implementation of the *Servient* that is enhanced with WT discovery mechanisms. It is worth remarking that the execution of the *Servient* requires significant computational resources that may not be afforded by constrained devices. To fill such gap, Sciullo *et al.* [133] propose a Micro Servient, *i.e.*, a W3C-compliant *Servient* that is supported by micro-controllers although with reduced functionalities (*e.g.* it can only expose WTs). A similar approach is also followed by [63], where the authors propose an automatic generator of WoT Servients in C++ with CoAP as communication capability for context-based applications.

3) MANAGEMENT TOOLS

We generically refer to a *Management* tool as a software that supports the adoption of the WoT paradigm and eases the development of new WoT functionalities starting from an existing IoT scenario. This means, for instance, providing support for the mapping of Things to WTs and enabling remote control and monitoring of the WTs through dashboards. Regarding the latter, Hautte *et al.* [58] propose a framework able to generate dynamic dashboarding applications from WTs provided with the Web Thing Model [147] (see Section IV-A). More in detail, their framework offers three main functionalities: a sensor discovery mechanism, a semantic matching system for sensors and visualizations, and a runtime system for continuously updating the dashboards. Similar automatic dashboarding functionalities for

FIGURE 7. The logical flow followed for the taxonomy of the tools analyzed.

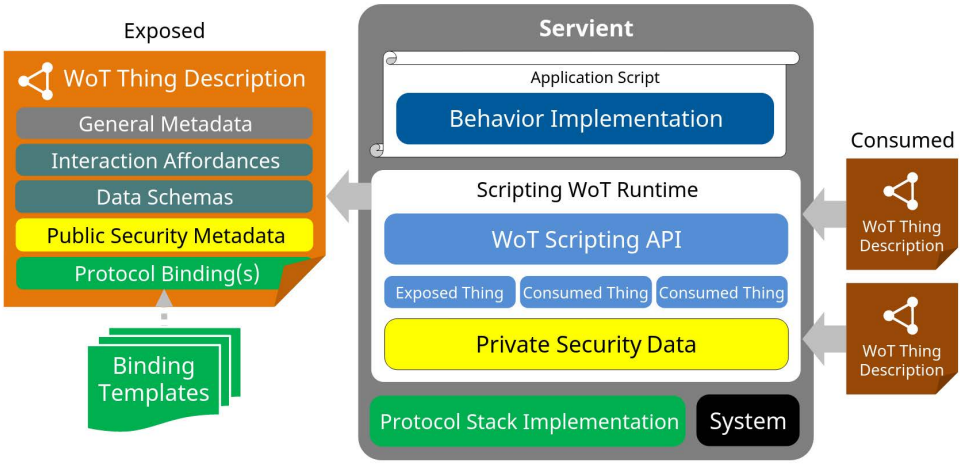


FIGURE 8. Implementation of a servient using the WoT scripting API [155].

W3C-compliant WTs are also offered by the previously cited WoT Store [128], [130], and its extensions for IoT monitoring applications [3], [162]; in addition, the tool provides a WoT ecosystem for the integrated management of WoT applications and data. Taking advantage of the *Node-RED*[6] platform, Ji *et al.* [67] propose a tool to integrate IoT devices, REST APIs, and WTs: more specifically, they developed a specific module of *Node-RED* capable of hosting a *Servient*. Through such integration, *Node-RED* can be turned into a complete mashup platform for the WoT. Finally, we cite the meta-platform in [90] which enables an Experiment-As-A-Service (EaaS) paradigm for IoT/WoT experiments. The tool supports the federation of heterogeneous IoT testbeds from the Fiesta-IoT project [1] thanks to the specification of a common Testbed API and the usage of a shared IoT registry [89] enabling the exchange of information in a semantically annotated format.

VI. DEPLOYMENTS

Despite the concept of WoT has been around for nearly two decades, we are witnessing a slow but steady adaptation that, in fact, encompasses nearly the same fields in which the IoT is used nowadays. In particular, this Section is devoted to gather major real-world deployments that rely upon the WoT – in all its facets – in order to show a glimpse of how and where it is used. We believe that our approach reflects the global trend in the adoption of the WoT at the time of writing, although listing a reduced set of deployments. First of all, along the Section we group use case studies based on their application field, which include some well-established vertical markets of the IoT (such as Home Automation, Smart Cities and Healthcare) as well as emerging applications (such as SHM) upon which the WoT has been particularly implied. We briefly report on each use case and, finally, we discuss possible trends and expectations. Table 4 summarizes the features of the studies taken into account.

A. SMART HOME

Home-sized contexts were used since the beginning of the WoT to provide the very first application samples, like in the paper of Heuer *et al.* [60]. In [22] an edge virtualization system for a Smart Home is presented. The building blocks of the architecture follow the W3C WoT specifications and the guidelines of the oneM2M technology are also integrated. In [144] a testbed for a Smart Home is presented. They integrate a sensor measuring PM10 and PM2.5 and implement an air quality monitoring cycle through actuators that control windows, all supervised by a custom WoT gateway.

[6]https://nodered.org/

TABLE 4. Major WoT deployments, in chronological order.

Study	Year	TRL	Place	Field	WoT Paradigm
Ruppen et al. [122]	2012	Academic	Switzerland	Healthcare	Legacy WoT
Corredor et al. [31]	2014	Academic	Spain	Healthcare	Legacy WoT
Paganelli et al. [112]	2016	Industrial	Europe	Smart City	Legacy WoT
Karim et al. [73]	2017	Academic	Tunisia	Smart Agriculture	Legacy WoT
Chakraborty and Datta [22]	2017	Academic	India	Smart Home	W3C WoT
El Jaouhari et al. [36]	2017	Academic	France	Healthcare	Legacy WoT
Thuluva et al. [145]	2017	Industrial	Germany	Industry 4.0	W3C WoT
Patel et al. [113]	2018	Academic	USA	Industry 4.0	SWoT
Klotz et al. [80]	2018	Industrial	Germany	Automotive	W3C WoT
Ji et al. [67]	2018	Academic	South Korea	Env. Monitoring	W3C WoT
Tobarra et al. [146]	2019	Academic	Spain	Education	Legacy WoT
Sciullo et al. [129]	2019	Academic	Italy	Industry 4.0	W3C WoT
Terius-Padron et al. [144]	2020	Academic	Spain	Smart Home	W3C WoT
Bauer et al. [11]	2020	Industrial	Germany	Smart Home	Mozilla WT
Sciullo et al. [130]	2020	Industrial	Germany, Italy	Industry 4.0	W3C WoT
Collins et al. [30]	2021	Academic	UK	Industry 4.0	W3C WoT
Aguzzi et al. [3]	2021	Industrial	Italy	SHM	W3C WoT
Ibaseta et al. [62]	2021	Industrial	Spain, Italy	Smart City	W3C WoT
Zyrianoff et al. [163]	2021	Academic	Brazil, Italy	Smart Agriculture	W3C WoT

Finally, we also cite the ongoing effort, within the ForeSight project, in integrating AI-based Smart Living platforms, such as OpenHAB, with the WoT at the control layer [11].

B. SMART CITIES & ENVIRONMENTAL MONITORING

The creation of a Smart City certainly represents one of the most ambitious and interesting goals in the world of IoT. Its complexity lies in the management of a very high number of heterogeneous sensors that need a uniform access interface. Moreover, it is essential to find a way to make data accessible and meaningful both for the end users and for other software agents and stakeholders.

Within the EU project SmartSantander [124], it has been proven how a WoT-based framework is an effective solution for modeling and turning smart things into web-addressable resources accessible through a REST interface. The proposed tool allows users to browse sensors, create virtual sensors, and register them in the system. It is also possible to create Web resources representing Points of Interest (PoIs), which can be seen as collections of descriptions, location-related data and sensors of a specific place. Both the virtual sensors and PoIs can have different types of permissions, and users can easily share these resources with each other [112].

In the domain of Smart Cities, we also register some interest towards ancillary services that intervene when particular unexpected events occur on a large scale. A crucial issue of smart city is the energy efficiency in buildings, currently tackled by Building Energy Management Systems (BEMS), which are coordinated systems of sensors and actuators that concur in optimizing the energy consumption in large edifices. In [62], the authors propose a BEMS based on energy retrofit using the W3C WoT within the scope of the HEART EU project [59], where a complete real-world case study is deployed. Finally, we cite the small-case testbed in [67] related to multi-sensor IoT environmental monitoring through the W3C WoT.

C. SMART AGRICULTURE

Smart agriculture includes a huge range of solutions aimed at improving crop production. Sensors are widely used to measure meteorological, (*i.e.* temperature, wind, sunshine) and soil (*i.e.* nutrients, moisture content, pH) conditions and such information can be wirelessly transmitted to edge nodes or directly to the cloud by several M2M communication technologies and infrastructures [101].

In their work, Karim *et al.* [73] present an alert system for plant water control based on a three-tier IoT/WoT architecture. A Web platform has been developed to monitor the operating conditions and data produced by sensors, and to allow the farmer to configure a series of alerts in case certain thresholds are exceeded. In this context, it is also worth mentioning the SWAMP project [139], which puts considerable research efforts towards using IoT technologies for water management and smart irrigation. In [163], the authors compare the usage of the FIWARE platform and the W3C WoT regarding the interoperability support and demonstrate a possible integration among the two approaches; in addition, they evaluate the scalability of both solutions on a large-scale testbed composed of soil moisture sensors provided by the SWAMP project and mapped to the W3C WoT.

D. STRUCTURAL HEALTH MONITORING

Structural Health Monitoring (SHM) deals with the continuous observation and analysis of civil and industrial structures (*e.g.* buildings, infrastructures, equipment etc), to prevent possible damages, or simply to optimize their maintenance over time. The most recent SHM systems take advantage of WoT technologies to enhance the data acquisition, management, and visualization.

In [3], [162], the authors describe the architecture and related implementation of the MODRON framework for sensor-to-cloud data acquisition and processing in SHM scenarios. Given the heterogeneity of sensors and protocols used

for structural monitoring, the proposed framework relies on the W3C WoT to abstract as much as possible from the sensing technologies. Its layered architecture includes an edge component exposing the WTs associated with physical (*i.e.* sensors) and virtual entities (*i.e.* monitored structure), and a cloud component that interacts with the WTs and implements distributed data storage, aggregation, and visualization.

E. AUTOMOTIVE

As time goes by, the automotive market demands more and more innovation which translates into the combination of systems such as the vehicle, infrastructure back-ends, and external data sources. These systems grow in complexity and autonomy, requiring a uniform way to understand each other and exchange information. In the paper [80] the authors use semantic technologies for enriching signal data in the automotive industry, specifically, they provide two main contributions: an ontology for standardize data definition for vehicle signals; the design and implementation of a W3C (Car) WT with a specific protocol binding with the LwM2M protocol.

F. INDUSTRY 4.0

Industry 4.0 is an evolution of traditional industry that aims to automate classic production practices using modern smart technologies, many of them related to big-data and the IoT. In this specific context, we often have to deal with devices and machinery that are not always up-to-date and that support different languages and protocols. Again, the use of WoT as an interoperability layer proves to be a winning solution, capable of abstracting this complexity and providing a homogeneous interface on which to build more complex applications and systems. An Industry 4.0 testbed is presented in [145], where they demonstrate the automation in the processes of on-boarding of a new device as well as low-effort re-engineering through semantics in an industrial scenario. In [113], authors propose a platform, largely based on the concept of SWoT, to bridge the gaps in Industry 4.0 scenarios that may still be present using legacy frameworks like RAMI 4.0 and IIRA. In particular, the proposed platform, namely SWeTI, is specifically oriented to tackle three major industrial challenges: (*i*) deep integration, which includes data heterogeneity from different domains, (*ii*) horizontal integration, which includes the interaction of different actors, and (*iii*) autonomous operations, in order to reduce manual effort in organization, manufacturing and fault tolerance. A more concrete testbed is presented in [132], where the W3C WoT standard is fully unleashed. Here, the authors show how to map heterogeneous sensing infrastructures based on different M2M technologies (WiFi, BLE, Zigbee) to the W3C WoT. In addition, they establish various Mash-up applications, based on ML techniques, able to orchestrate the sensing operations from the scenario, in order to maximize system-level requirements, such as the energy efficiency or the throughput. Differently, in [129] an industrial testbed on production lines is deployed. In this work, production lines work by means of deterministic networks, which are entirely orchestrated via WoT technologies that interact via the machine-specific communication protocols. Lastly, we also report the recent work in [30], where the W3C WoT is leveraged for remotely controlling a set of scientific instrumentation, such as microscopes.

G. HEALTHCARE

The advent of smart devices has opened new possibilities in the Healthcare domain. In particular, recent developments in wireless technologies, sensors, and actuators are enabling new classes of healthcare applications ranging from patient monitoring to high level data analysis and sharing. Issues such as interoperability and security of these systems have even more radical importance in this context compared to others, as we are dealing with medical data that is often severely restricted due to privacy concerns given by country and area-specific legal frameworks. For these reasons, the WoT can become the reference point for compatibility between these systems, providing secure access mechanisms through consolidated Web standards. An early work [122] envisions the usage of the Legacy WoT in order to automatize processes that were largely done on paper. In this case, the scenario features a pool of patients taking different exams and a pool of caregivers that need to be notified promptly of the exam result. The WoT here acts as a blackbox where exam reports are distributed with different policies and patients and caregivers are seen as Web resources. More recently, architectures for similar use cases were proposed, such as in [36], where authors envision the joint usage of the WoT and WebRTC. This enables patients to monitor their physical conditions and to have real-time interactions with their medical referent. In Corredor *et al.* [31], a WoT-based platform is proposed to facilitate deployments of enriched and heterogeneous e-health smart spaces: ecosystems of embedded sensor and actuator devices deployed on the user's environment. The system is extremely lightweight and allows resource-constrained devices to be used as smart gateways.

H. EDUCATION

In the context of education, we often witness the difficulties of distance learning, recently exacerbated by the outbreak of the COVID-19. Distance learning becomes critical when the interaction with physical things and hardware is necessary for scholars. The WoT paradigm here comes as a timely helping hand, especially in the context of programmable boards. It is the example of the WoT platform presented in [146], where programmable IoT devices are abstracted into a virtual laboratory where they can be remotely programmed and orchestrated, as well as monitored through dashboards.

I. DISCUSSION

As it can be seen from Table 1, field studies that make use of the WoT start to appear at the beginning of the 2010's and grow consistently in numbers after 2015. This suggests that the WoT is gaining interest and, if the trend stays steady, it is expected to have a larger audience over the next years.

Another point that looks quite clear is the predominance of the implementations that rely on the W3C WoT standard in the very latest years, in contrast with the Legacy WoT, which was more popular earlier. This observation is further strengthened by the recent endorsement of the W3C WoT by other standardization bodies. A third remark is that the majority of the works are European. This could be justified by the number of EU projects [59], [124], [139] where the usage of the WoT has been reported. On the other hand, we also note that use case studies are less likely to be adopted in industrial scenarios to date, in contrast with academic testbeds. This is expected because the standard has not yet reached the necessary maturity to meet the market offer; however, the amount of investments by project bodies highlights that the way to high TRL (Technology Readiness Level) is being paved.

VII. RESEARCH DIRECTIONS

The WoT landscape opens up several opportunities for research works. In Section IV we presented the main building blocks of the WoT, which originated as many research areas. From there, we can observe that several near-future improvements are deeply discussed within and outside the standardization entities. At the same time, the current IoT landscape is characterized by novel technologies, emerging from the continuous hybridization of the existing ones, that may also benefit from a WoT paradigm. In this Section, we aim to outline some research directions that may open the way to next-generation WoT systems. Some of these directions are built on top of the latest progress in the context of the pure Web and the IoT (*e.g.* the Digital Twin concept). Others are more visionary and involve the integration between the WoT and novel technologies, such as ML and the Blockchain, in order to respond to future market demand.

A. WOT AUTOMATION

Although the WoT has already given a significant boost to interoperability in IoT contexts, we are still far from an ecosystem able to deal with their variability. Indeed, IoT contexts are rarely static, and, due to their increasing numbers [29], human intervention is no longer an option. Many industrial and academic realities are shifting their attention towards valorizing the automation of Systems-of-Systems along their engineering lifecycle [86]. In such an environment, infrastructures that enable the automatic deployment of WTs and their planning are highly valuable, since the on-boarding process is a costly and time-consuming task for humans. Some automated deployments of WTs have been proposed based on a number of design-time modeling standards and tools. One example is UML [74], but we may envision several others (*e.g.* SysML, ADL, Petri-nets, etc.). However, the reality of IoT deployments does not guarantee to always have a structured representation of a system at hand, therefore pure translation from a modeling language to, *e.g.*, a set of TDs may not be enough. In fact, the future landscape will need to involve inference methods, for instance based on ML techniques, to cope with such heterogeneity and, at the same time, minimize the human effort to make the ecosystem self-adaptable.

B. DIGITAL WEB-TWIN

A ''Digital Twin'' (DT) is a near-real-time digital virtualization of a physical Thing. The concept was introduced by [47] where the virtual representation was limited to an enriched representation of the physical counterpart and the two entities were virtually indistinguishable. In the last years, the DT embodied new features and started to become an essential and crucial element in many industrial processes due to its ability of monitoring and forecasting. The DT, in fact, can be used to predict the future behavior of the virtualized Thing (used for fault diagnosis and prognosis) as well as to explore ''virtual subspaces'' executing the *what-if* analysis in order to anticipate unknown behaviors. Given the relevance of the concept, the DT concept has already been included in the W3C IG's *Use Cases and Requirements* notes [88]. Due to the well-defined structure of a WT, the advent of the Digital Web-Twin (DWT) will bring the DT deployment to a new era, *e.g.* enabling self-optimization capabilities [6] and moving from a scenario composed of separate DTs to networks of integrated, interacting DTs able to reproduce the dynamics of a complex system.

C. BLOCKCHAIN-ENABLED WOT

With the term Blockchain, we no longer refer exclusively to Bitcoin and cryptocurrencies in general, but to a set of technologies that are progressively changing several sectors such as finance, healthcare, cybersecurity, IoT, and more. These technologies are increasingly used to solve secure data storage problems thanks to their decentralized architecture and data immutability. They also provide full availability of their services (*e.g.* Smart Contracts on Ethereum[7]) and several mechanisms for privacy and digital identity management. Such features find a very good spot in the Web ecosystem and in fact, we can already find several studies that try to combine the Semantic Web with Blockchain technologies in an attempt to create a powerful synergy that enhances both worlds [21]. This growing interest is only at the beginning and leaves room for strong future research and experimentation, specifically, the WoT could find a perfect spot by serving as an access layer to the IoT world for these technologies. An interesting topic might involve the design of a WoT-based Data Oracle capable of retrieving data from heterogeneous devices (*e.g.* some temperature sensors) on behalf of the Smart Contracts that need it. Conversely, Blockchain technologies could be used to ensure the identity and therefore the trustability of WTs and to enable their secure and traceable communication in a scenario of automated Thing2Thing interactions. It is important to note that Blockchain technologies are only a subset of a large group of Distributed Ledger Technologies (DLTs).

[7]https://ethereum.org/en/

In this broader group are projects such as IOTA[8] and Hedera[9] that exploit mechanisms that allow them to achieve a high number of transactions per second. These features make them prime candidates to handle the high number of interactions in a hypothetical Thing2Thing distributed communication scenario.

D. WEB-THING-AS-A-SERVICE (WTAAS)

The IoT-as-a-Service (IoTaaS) platform [19] was established from the intersection between the Cloud computing and the IoT paradigm and identifies a new ecosystem where IoT devices can host one or multiple services that can be requested and executed by third-party IoT applications (called IoT tenant applications). The deployment of such a system can benefit from the WoT architecture in multiple ways. The IoTaaS is, in fact, platform-specific and its implementation still suffers from the fragmentation of the IoT. The natural evolution of this concept is the Web-Things-as-a-Service (WTaaS), where the components are mapped to WTs. The WTaaS paradigm can provide flexibility by coping with the heterogeneity of software platforms and devices and enabling service discovery, scaling, and composition of services. In addition, it can guarantee the complete separation among the deployed platforms and services (the Control Plane), and the tenant applications (the User Plane). On the other side, the WoT tenant applications using the WTaaS paradigm may ask for computational and network resources, which are managed by the Network Operators in a seamless way. Similar to the 5G Network Slicing concept [41], the available hardware can be shared among multiple WoT tenant applications by slicing the available physical/virtual resources based on the tenants' QoS requests. In this context, the dynamic allocation of WTs to computational nodes is of paramount importance due to the dynamicity of workloads. Aguzzi *et al.* [2] is a pioneering work that explores such a possibility by introducing the Migratable WoT (M-WoT) where the services are deployed using the WoT paradigm, supporting dynamic orchestration and stateful migration of WTs in a cloud-edge continuum.

VIII. CONCLUSION

In this paper, we have analyzed the WoT landscape that spans two decades by now. The content of our study reflects the joint effort of academia and standardization bodies in proposing an aligned solution that aims to solve the age-old interoperability problem in the context of IoT. More in detail, we first looked at the history of the WoT, by identifying the major milestones and current trends that characterized this paradigm. In addition, we discussed few fundamental definitions (*e.g.* Web Thing, Mashup Application, etc.) that form a common ground for all such research branches. Subsequently, we framed the current state of the research by providing a taxonomy over both architectural styles and the building blocks in the WoT, also categorizing existing literature into each of these areas. Moreover, we discussed the enabling technologies and maturity of the current vertical market, by looking at the most popular deployments and tools. Finally, we discussed the future of the WoT by identifying some major research directions. From the literature review, we can conclude that there is a clear interest of both academia and industry in the WoT, and that such interest is justified by the possibilities offered by the strict integration between the Web ecosystem and the IoT. Furthermore, it is also clear how the WoT may pave the way to multiple and novel application domains that are in line with emerging research trends of the IoT.

[8]https://www.iota.org
[9]https://hedera.com

REFERENCES

[1] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, and V. Issarny, "Unified IoT ontology to enable interoperability and federation of testbeds," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 70–75.
[2] C. Aguzzi, L. Gigli, L. Sciullo, A. Trotta, and M. Felice, "From cloud to edge: Seamless software migration at the era of the web of things," *IEEE Access*, vol. 8, pp. 228118–228135, 2020.
[3] C. Aguzzi, L. Gigli, L. Sciullo, A. Trotta, F. Zonzini, L. De Marchi, M. Di Felice, A. Marzani, and T. S. Cinotti, "MODRON: A scalable and interoperable web of things platform for structural health monitoring," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–7.
[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.
[5] 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.
[6] R. Anarfi and K. K. Fletcher, "A reinforcement learning approach to web API recommendation for mashup development," in *Proc. IEEE World Congr. Services (SERVICES)*, Jul. 2019, pp. 372–373.
[7] F. Antoniazzi and F. Viola, "Building the semantic web of things through a dynamic ontology," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10560–10579, Dec. 2019.
[8] Arrowhead Consortium. (2015). *Arrowhead—Ahead of the Future*. [Online]. Available: https://www.arrowhead.eu
[9] M. Baqer, "Enabling collaboration and coordination of wireless sensor networks via social networks," in *Proc. 6th IEEE Int. Conf. Distrib. Comput. Sensor Syst. Workshops (DCOSSW)*, Jun. 2010, pp. 1–2.
[10] P. Barnaghi and M. Presser, "Publishing linked sensor data," in *Proc. 3rd Int. Conf. Semantic Sensor Netw.*, Aachen, Germany, vol. 668, 2010, pp. 1–16.
[11] J. Bauer, M. Hechtel, C. Konrad, M. Holzwarth, H. Hoffmann, T. Feld, S. Schneider, I. Zinnikus, A. Mayr, and J. Franke, "Foresight—An AI-driven smart living platform, approach to add access control to openHAB," in *Proc. Int. Conf. Smart Homes Health Telematics*. Cham, Switzerland: Springer, 2020, pp. 432–440.
[12] Z. Benomar, F. Longo, G. Merlino, and A. Puliafito, "A Stack4Things-based web of things architecture," in *Proc. Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData), IEEE Congr. Cybermatics (Cybermatics)*, Nov. 2020, pp. 113–120.
[13] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Sci. Amer.*, vol. 284, no. 5, pp. 34–43, May 2001.
[14] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data—The story so far," *Int. J. Semantic Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, Jul. 2009.
[15] M. Blackstock and R. Lea, "IoT mashups with the WoTKit," in *Proc. 3rd IEEE Int. Conf. Internet Things*, Oct. 2012, pp. 159–166.
[16] M. Blank, H. Lahbaiel, S. Kaebisch, and H. Kosch, "Role models and lifecycles in IoT and their impact on the W3C WoT thing description," in *Proc. 8th Int. Conf. Internet Things*, Oct. 2018.
[17] H. Bornholdt, D. Jost, P. Kisters, M. Rottleuthner, D. Bade, W. Lamersdorf, T. C. Schmidt, and M. Fischer, "SANE: Smart networks for urban citizen participation," in *Proc. 26th Int. Conf. Telecommun. (ICT)*, Apr. 2019, pp. 496–500.
[18] T. Bray, Ed., *The JavaScript Object Notation (JSON) Data Interchange Format*, document RFC 8259, Internet Standard, Dec. 2017.

[19] A. A. Brincat, F. Pacifici, and F. Mazzola, "IoT as a service for smart cities and nations," *IEEE Internet Things Mag.*, vol. 2, no. 1, pp. 28–31, Mar. 2019.

[20] A. Bröring, S. Schmid, C.-K. Schindhelm, A. Khelil, S. Käbisch, D. Kramer, D. Le Phuoc, J. Mitic, D. Anicic, and E. Teniente, "Enabling IoT ecosystems through platform interoperability," *IEEE Softw.*, vol. 34, no. 1, pp. 54–61, Jan./Feb. 2017.

[21] J. Cano-Benito, A. Cimmino, and R. García-Castro, "Towards blockchain and semantic web," in *Business Information Systems Workshops*, W. Abramowicz and R. Corchuelo, Eds. Cham, Switzerland: Springer, 2019, pp. 220–231.

[22] T. Chakraborty and S. K. Datta, "Home automation using edge computing and Internet of Things," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, Nov. 2017, pp. 47–49.

[23] B. Cheng, S. Zhao, J. Qian, Z. Zhai, and J. Chen, "Lightweight service mashup middleware with REST style architecture for IoT applications," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 3, pp. 1063–1075, Sep. 2018.

[24] C.-M. Chituc, "Towards seamless communication in the web of things: Are standards sufficient to ensure interoperability?" in *Proc. 13th Int. Conf. Commun. (COMM)*, 2020, pp. 427–431.

[25] T. Chou, *Precision—Principles, Practices and Solutions for the Internet of Things*. New York, NY, USA: McGraw-Hill, 2017.

[26] T.-Y. Chung, I. Mashal, O. Alsaryrah, V. Huy, W.-H. Kuo, and A. P. Agrawal, "Social web of things: A survey," in *Proc. Int. Conf. Parallel Distrib. Syst.*, 2013, pp. 570–575.

[27] A. Ciortea, O. Boissier, A. Zimmermann, and A. M. Florea, "Reconsidering the social web of things. Position paper," in *Proc. UbiComp Adjunct-Adjunct Publication ACM Conf. Ubiquitous Comput.*, 2013, pp. 1535–1544.

[28] F. Cirillo, G. Solmaz, E. L. Berz, M. Bauer, B. Cheng, and E. Kovacs, "A standard-based open source IoT platform: FIWARE," *IEEE Internet Things Mag.*, vol. 2, no. 3, pp. 12–18, Sep. 2019.

[29] CISCO, "Cisco annual internet report (2018–2023)," Cisco, San Jose, CA, USA, Tech. Rep. C11-741490-01, 2020.

[30] J. T. Collins, J. Knapper, J. Stirling, S. McDermott, and R. Bowman, "Modern microscopy with the web of things: The openflexure microscope software stack," 2021, *arXiv:2101.00933*.

[31] I. Corredor, E. Metola, A. Bernardos, P. Tarrío, and J. Casar, "A lightweight web of things open platform to facilitate context data management and personalized healthcare services creation," *Int. J. Environ. Res. Public Health*, vol. 11, no. 5, pp. 4676–4713, Apr. 2014. [Online]. Available: https://www.mdpi.com/1660-4601/11/5/4676

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

[33] J. Delsing, *IoT Automation: Arrowhead Framework*. Boca Raton, FL, USA: CRC Press, 2017.

[34] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the Internet of Things," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Automat. (ETFA)*, Sep. 2015, pp. 1–8.

[35] M. C. Domingo, "A context-aware service architecture for the integration of body sensor networks and social networks through the IP multimedia subsystem," *IEEE Commun. Mag.*, vol. 49, no. 1, pp. 102–108, Jan. 2011.

[36] S. E. Jaouhari, A. Bouabdallah, J.-M. Bonnin, and T. Lemlouma, "Toward a smart health-care architecture using WebRTC and WoT," in *Proc. World Conf. Inf. Syst. Technol.* Cham, Switzerland: Springer, 2017, pp. 531–540.

[37] ETSI. (2017). *Smart Appliances Reference Ontology and oneM2M Mapping*. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103200_103299/103264/02.01.01_60/ts_103264v020101p.pdf

[38] ETSI. (2019). *CIM Information Model*. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/006/01.01.01_60/gs_CIM006v010101p.pdf

[39] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002.

[40] J. M. C. Fonseca and M. Bauer, "Position statement from ETSI ISG CIM," in *Proc. 2nd W3C Workshop Web Things*, 2020.

[41] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[42] B. Francis, "Web thing API," Tech. Rep., 2021. [Online]. Available: https://webthings.io/api/

[43] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, and K. Wasielewska, "Semantic interoperability in the Internet of Things: An overview from the INTER-IoT perspective," *J. Netw. Comput. Appl.*, vol. 81, pp. 111–124, Mar. 2017.

[44] A. G. Mangas and F. J. S. Alonso, "WOTPY: A framework for web of things applications," *Comput. Commun.*, vol. 147, pp. 235–251, Nov. 2019, doi: 10.1016/j.comcom.2019.09.004.

[45] O. Garcia-Morchon, S. Kumar, and M. Sethi, *Internet of Things (IoT) Security: State of the Art and Challenges*, document RFC 8576, Apr. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8576.txt

[46] A. Ghasempour, "Internet of Things in smart grid: Architecture, applications, services, key technologies, and challenges," *Inventions*, vol. 4, no. 1, p. 22, Mar. 2019.

[47] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," Florida Inst. Technol., Melbourne, FL, USA, White Paper, 2014, pp. 1–7, vol. 1.

[48] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable web of things," in *Proc. 8th IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM Workshops)*, Mar. 2010, pp. 702–707.

[49] D. Guinard, M. Mueller, and J. Pasquier-Rocha, "Giving RFID a REST: Building a web-enabled EPCIS," in *Proc. Internet Things (IOT)*, 2010, pp. 1–8, doi: 10.1109/IOT.2010.5678447.

[50] D. Guinard and V. Trifa, "Towards the web of things: Web mashups for embedded devices," in *Proc. Workshop Mashups, Enterprise Mashups Lightweight Composition Web (MEM)*, 2009, pp. 1–8.

[51] D. Guinard, V. Trifa, T. Pham, and O. Liechti, "Towards physical mashups in the web of things," in *Proc. 6th Int. Conf. Netw. Sens. Syst. (INSS)*, Jun. 2009, pp. 196–199.

[52] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Proc. Internet of Things (IOT)*, 2010, pp. 1–8, doi: 10.1109/IOT.2010.5678452.

[53] D. Guinard, C. A. Velasco, J. B. Domingue, and D. Carrera. (2015). *Submission Request to W3C: Web Thing Model*. [Online]. Available: https://www.w3.org/Submission/2015/01/

[54] D. D. Guinard and V. M. Trifa. (Apr. 2008). *What is the Web of Things? ApacheCon Europe, Presentation*. [Online]. Available: https://webofthings.org/2017/04/08/what-is-the-web-of-things/

[55] D. D. Guinard and V. M. Trifa, *Building the Web of Things*, vol. 3. Shelter Island, NY, USA: Manning Publications, 2016.

[56] 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 World Forum Internet Things (WF-IoT)*, Jan. 2018, pp. 86–91.

[57] K. Hartke, "The constrained restful application language (CoRAL)," Internet Eng. Task Force (IETF), Fremont, CA, USA, Tech. Rep. draft-hartke-t2trg-coral-04, 2017, p. 47.

[58] S. V. Hautte, P. Moens, J. Van Herwegen, D. De Paepe, B. Steenwinckel, S. Verstichel, F. Ongenae, and S. Van Hoecke, "A dynamic dashboarding application for fleet monitoring using semantic web of things technologies," *Sensors*, vol. 20, no. 4, p. 1152, Feb. 2020.

[59] *HEART Eu Project*, 2020.

[60] J. Heuer, J. Hund, and O. Pfaff, "Toward the web of things: Applying web technologies to the physical world," *Computer*, vol. 48, no. 5, pp. 34–42, 2015.

[61] P. Hitzler, "A review of the semantic web field," *Commun. ACM*, vol. 64, no. 2, pp. 76–83, Jan. 2021.

[62] D. Ibaseta, A. García, M. Álvarez, B. Garzón, F. Díez, P. Coca, C. D. Pero, and J. Molleda, "Monitoring and control of energy consumption in buildings using WoT: A novel approach for smart retrofit," *Sustain. Cities Soc.*, vol. 65, Feb. 2021, Art. no. 102637.

[63] M. Iglesias-Urkia, A. Gómez, D. Casado-Mansilla, and A. Urbieta, "Automatic generation of web of things servients using thing descriptions," *Pers. Ubiquitous Comput.*, pp. 1–17, Jul. 2020, doi: 10.1007/s00779-020-01413-3.

[64] A. J. Jara, A. C. Olivieri, Y. Bocchi, M. Jung, W. Kastner, and A. F. Skarmeta, "Semantic web of things: An analysis of the application semantics for the IoT moving towards the IoT convergence," *Int. J. Web Grid Services*, vol. 10, nos. 2–3, pp. 244–272, 2014.

[65] S. Javaid, H. Afzal, F. Arif, and N. Iltaf, "Trust management for SOA based social WoT system," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 387–392, doi: 10.23919/ICACT.2018.8323767.

[66] C. Jennings, Z. Shelby, J. Arkko, A. Keränen, and C. Bormann, *Sensor Measurement Lists (SenML)*, document RFC 8428, Aug. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8428.txt

[67] Y. Ji, K. Ok, and W. S. Choi, "Web of things based IoT standard interworking test case," in *Proc. 5th Conf. Syst. Built Environ.*, Nov. 2018pp. 182–183.
[68] Y. Jung and A. Causey, "HyDRS-WoT: Hybrid disaster response system using web of things," in *Proc. 21st Int. Conf. Inf. Integr. Web-Based Appl. Services*, 2019, pp. 10–13.
[69] S. Käbisch, T. Kamiya, M. McCool, V. Charpenay, and A. Kovatsch. Web of things (WoT) thing description. W3C Recommendation, Apr. 2020. [Online]. Available: https://www.w3.org/TR/wot-thing-description
[70] A. Kamilaris, K. Papakonstantinou, and A. Pitsillides, "Exploring the use of DNS as a search engine for the web of things," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Mar. 2014, pp. 100–105.
[71] A. Kamilaris and A. Pitsillides, "Social networking of the smart home," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2010, pp. 2632–2637.
[72] A. Kamilaris, S. Yumusak, and M. I. Ali, "WOTS2E: A search engine for a semantic web of things," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 436–441.
[73] F. Karim, F. Karim, and A. frihida, "Monitoring system using web of things in precision agriculture," *Proc. Comput. Sci.*, vol. 110, pp. 402–409, Jan. 2017.
[74] A. Kast, E. Korkan, S. Käbisch, and S. Steinhorst, "Web of things system description for representation of mashups," in *Proc. Int. Conf. Omni-Layer Intell. Syst. (COINS)*, Aug. 2020, pp. 1–8.
[75] A. Kast, E. Korkan, S. Käbisch, and S. Steinhorst, "Web of things system description for representation of mashups," in *Proc. Int. Conf. Omni-Layer Intell. Syst. (COINS)*, Aug. 2020, pp. 1–8.
[76] G. Kellogg, P.-A. Champin, and D. Longley. (May 2020). *JSON-LD 1.1. W3C Proposed Recommendation*. [Online]. Available: https://www.w3.org/TR/json-ld11/
[77] M. Kelly, "JSON hypertext application language," Internet Eng. Task Force (IETF) Draft, Fremont, CA, USA, Tech. Rep. kelly-json-hal-08, May 2016, p. 11. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-kelly-jsonhal-08
[78] A. Keränen, F. M. Kovatsch, and K. Hartke. (Feb. 2021). RESTful design for Internet of Things systems. Internet Engineering Task Force. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-rest-iot-07
[79] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic, "People, places, things: Web presence for the real world," in *Proc. 3rd IEEE Workshop Mobile Comput. Syst. Appl.*, Los Alamitos, CA, USA, Dec. 2000, pp. 19–28, doi: 10.1109/MCSA.2000.895378.
[80] B. Klotz, S. K. Datta, D. Wilms, R. Troncy, and C. Bonnet, "A car as a semantic web thing: Motivation and demonstration," in *Proc. Global Internet Things Summit (GIoTS)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Nov. 2018, pp. 1–6.
[81] E. Korkan, H. B. Hassine, V. E. Schlott, S. Käbisch, and S. Steinhorst, "WoTify: A platform to bring web of things to your devices," 2019, *arXiv:1909.03296*.
[82] E. Korkan, S. Kaebisch, M. Kovatsch, and S. Steinhorst, "Safe interoperability for web of things devices and systems," in *Languages, Design Methods, and Tools for Electronic System Design*. Cham, Switzerland: Springer, 2020, pp. 47–69.
[83] E. Korkan, F. Salama, S. Kaebisch, and S. Steinhorst, "A-MaGe: Atomic mashup generator for the web of things," in *Web Engineering*, M. Brambilla, R. Chbeir, F. Frasincar, I. Manolescu, Eds. Cham, Switzerland: Springer, 2021, pp. 320–327.
[84] M. Koster, "Media types for hypertext sensor markup (HSML)," Internet Eng. Task Force (IETF), Fremont, CA, USA, Tech. Rep. draft-koster-t2trg-hsml-01, Mar. 2017, p. 49. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-koster-t2trg-hsml-01
[85] P. Krawiec, M. Sosnowski, J. M. Batalla, C. X. Mavromoustakis, G. Mastorakis, and E. Pallis, "Survey on technologies for enabling real-time communication in the web of things," in *Beyond the Internet of Things*. Cham, Switzerland: Springer, 2017, pp. 323–339.
[86] G. Kulcsár, P. Varga, S. M. Tatara, F. Montori, A. Michel Iñigo, G. Urgese, and P. Azzoni, "Modeling an industrial revolution: How to manage large-scale, complex IoT ecosystems?" in *Proc. 2nd Workshop Manage. Ind. (MFI), IFIP/IEEE IM-Int. Symp. Integr. Netw. Manage.*, May 2021, pp. 323–339.
[87] N. Kumar and S. Ahmad, "Security threats in layered architecture of web of things," in *Proc. 4th Int. Conf. Inventive Syst. Control (ICISC)*, Jan. 2020, pp. 745–750.
[88] M. Lagally. (2021). *W3C—Digital Twin Use Case*. [Online]. Available: https://www.w3.org/TR/2021/NOTE-wot-usecases-20210518/#digital-twin
[89] J. Lanza, L. Sánchez, D. Gómez, J. R. Santana, and P. Sotres, "A semantic-enabled platform for realizing an interoperable web of things," *Sensors*, vol. 19, no. 4, p. 869, Feb. 2019.
[90] J. Lanza, L. Sanchez, J. R. Santana, R. Agarwal, N. Kefalakis, P. Grace, T. Elsaleh, M. Zhao, E. Tragos, H. Nguyen, F. Cirillo, R. Steinke, and J. Soldatos, "Experimentation as a service over semantically interoperable Internet of Things testbeds," *IEEE Access*, vol. 6, pp. 51607–51625, 2018.
[91] K.-H. Le, S. K. Datta, C. Bonnet, and F. Hamon, "WoT-AD: A descriptive language for group of things in massive IoT," in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, Apr. 2019, pp. 257–262.
[92] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. 11th IEEE Int. Symp. Object Compon.-Oriented Real-Time Distrib. Comput. (ISORC)*, May 2008, pp. 363–369.
[93] E. Lee, Y.-D. Seo, S.-R. Oh, and Y.-G. Kim, "A survey on standards for interoperability and security in the Internet of Things," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1020–1047, 2nd Quart., 2021.
[94] W. Li, G. Tropea, A. Abid, A. Detti, and F. Gall, "Review of standard ontologies for the web of things," in *Proc. Global IoT Summit (GIoTS)*, 2019, pp. 1–6.
[95] L. Mainetti, V. Mighali, and A. L. Patrono, "A software architecture enabling the web of Things," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 445–454, Jun. 2015.
[96] N. Mangra and A. Ghasempour. (2018). *Smart Cities: Connected Ecosystem of Ecosystems, IEEE Perspectives on 5G Applications and Services*. [Online]. Available: https://futurenetworks.ieee.org/images/files/pdf/applications/Smart-Cities_030518.pdf
[97] J. A. Martins, A. Mazayev, and N. Correia, "Hypermedia APIs for the web of things," *IEEE Access*, vol. 5, pp. 20058–20067, 2017.
[98] S. S. Mathew, Y. Atif, Q. Z. Sheng, and Z. Maamar, "Web of things: Description, discovery and integration," in *Proc. Int. Conf. Internet Things, 4th Int. Conf. Cyber, Phys. Social Comput.*, Oct. 2011, p. 9–15.
[99] M. McCool and E. Reshetova, "Distributed security risks and opportunities in the W3C web of things," in *Proc. Workshop Decentralized IoT Secur. Standards (DISS)*, San Diego, CA, USA, Feb. 2018. [Online]. Available: http://www.ndss-symposium.org, doi: 10.14722/diss.2018.23008.
[100] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, in *The Internet of Things: Mapping the Value Beyond the Hype*, vol. 24. New York, NY, USA: McKinsey Global Institute, 2015.
[101] F. Montori, L. Bedogni, M. Di Felice, and L. Bononi, "Machine-to-machine wireless communication technologies for the Internet of Things: Taxonomy, comparison and open issues," *Pervasive Mobile Comput.*, vol. 50, pp. 56–81, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574119217303668
[102] M. Mrissa, L. Médini, J.-P. Jamont, N. Le Sommer, and J. Laplace, "An avatar architecture for the web of things," *IEEE Internet Comput.*, vol. 19, no. 2, pp. 30–38, Mar./Apr. 2015, doi: 10.1109/MIC.2015.19.
[103] I. Nadim, Y. Elghayam, and A. Sadiq, "Semantic discovery architecture for dynamic environments of web of things," in *Proc. Int. Conf. Adv. Commun. Technol. Netw. (CommNet)*, Apr. 2018, pp. 1–6.
[104] B. Negash, T. Westerlund, and H. Tenhunen, "Towards an interoperable Internet of Things through a web of virtual things at the fog layer," *Future Gener. Comput. Syst.*, vol. 91, pp. 96–107, Feb. 2019.
[105] A. D. Norman, *The Design of Everyday Things*. New York, NY, USA: Basic Books, 2002.
[106] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in Internet of Things: Taxonomies and open challenges," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 796–809, 2019, doi: 10.1007/s11036-018-1089-9.
[107] M. Noura and M. Gaedke, "WoTDL: Web of things description language for automatic composition," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Oct. 2019, pp. 413–417.
[108] M. Noura, A. Gyrard, S. Heil, and M. Gaedke, "Automatic knowledge extraction to build semantic web of things applications," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8447–8454, Oct. 2019.
[109] O. Novo and M. D. Francesco, "Semantic interoperability in the IoT," *ACM Trans. Internet Things*, vol. 1, no. 1, pp. 1–25, Mar. 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3375838
[110] oneM2M Consortium. (2018). *TS-0012 oneM2M Base Ontology*. [Online]. Available: https://git.onem2m.org/MAS/BaseOntology
[111] Open Connectivity Foundation. (2022). *OCF Core Specification V2.2.25*. [Online]. Available: https://openconnectivity.org/specs/OCF_Core_Specification_v2.2.5.pdf
[112] F. Paganelli, S. Turchi, and D. Giuli, "A web of things framework for RESTful applications and its experimentation in a smart city," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1412–1423, Dec. 2016.

[113] P. Patel, M. I. Ali, and A. Sheth, "From raw data to smart manufacturing: AI and semantic web of things for industry 4.0," *IEEE Intell. Syst.*, vol. 33, no. 4, pp. 79–86, Jul. 2018.
[114] B. Peterson and B. Vogel, "Prototyping the Internet of Things with web technologies: Is it easy?" in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2018, pp. 518–522.
[115] A. Pintus, D. Carboni, and A. Piras, "Paraimpu: A platform for a social web of things," in *Proc. 21st Int. Conf. Companion World Wide Web (WWW Companion)*, 2012, pp. 401–404.
[116] M. Presser, P. M. Barnaghi, M. Eurich, and C. Villalonga, "The SENSEI project: Integrating the physical world with the digital world of the network of the future," *IEEE Commun. Mag.*, vol. 47, no. 4, pp. 1–4, Apr. 2009.
[117] D. Raggett, "Compose: An open source cloud-based scalable IoT services platform," *ERCIM News*, vol. 101, pp. 30–31, Apr. 2015.
[118] D. Raggett, "The web of things: Challenges and opportunities," *Computer*, vol. 48, no. 5, pp. 26–32, May 2015. [Online]. Available: https://www.w3.org/community and https://www.dali-ag.org
[119] M. A. Rahman, A. El Saddik, and W. Gueaieb, "Data visualization: From body sensor network to social networks," in *Proc. IEEE Int. Workshop Robot. Sensors Environ.*, Nov. 2009, pp. 157–162.
[120] A. Rhayem, M. B. A. Mhiri, and F. Gargouri, "Semantic web technologies for the Internet of Things: Systematic literature review," *Internet Things*, vol. 11, Sep. 2020, Art. no. 100206. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660520300421, doi: 10.1016/j.iot.2020.100206.
[121] L. Roffia, P. Azzoni, C. Aguzzi, F. Viola, F. Antoniazzi, and T. S. Cinotti, "Dynamic linked data: A SPARQL event processing architecture," *Future Internet*, vol. 10, no. 4, p. 36, Apr. 2018.
[122] A. Ruppen, J. Pasquier, J.-F. Wagen, B. Wolf, and R. Guye, "A WoT approach to eHealth: Case study of a hospital laboratory alert escalation system," in *Proc. 3rd Int. Workshop Web Things (WOT)*, 2012, pp. 1–6.
[123] M. Ruta, F. Scioscia, and E. Di Sciascio, "Enabling the semantic web of things: Framework and architecture," in *Proc. IEEE 6th Int. Conf. Semantic Comput. (ICSC)*, Sep. 2012, pp. 345–347.
[124] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, and R. Ramdhany, "SmartSantander: IoT experimentation over a smart city testbed," *Comput. Netw.*, vol. 61, pp. 217–238, Mar. 2014.
[125] SANE. (2021). *SANE Web of Things Servient*. [Online]. Available: https://github.com/sane-city/wot-servient
[126] R. Sardar and T. Anees, "Web of things: Security challenges and mechanisms," *IEEE Access*, vol. 9, pp. 31695–31711, 2021.
[127] T. Schmid and M. B. Srivastava, "Exploiting social networks for sensor data sharing with senseshare," Center Embedded Netw. Sens., Univ. California, Los Angeles, CA, USA, 2007. [Online]. Available: https://escholarship.org/uc/item/4919w4vh
[128] L. Sciullo, C. Aguzzi, M. Di Felice, and T. S. Cinotti, "WoT store: Enabling things and applications discovery for the W3C web of things," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–8.
[129] L. Sciullo, S. Bhattacharjee, and M. Kovatsch, "Bringing deterministic industrial networking to the W3C web of things with TSN and OPC UA," in *Proc. 10th Int. Conf. Internet Things*, Oct. 2020, pp. 1–8.
[130] L. Sciullo, L. Gigli, A. Trotta, and M. D. Felice, "WoT store: Managing resources and applications on the web of things," *Internet Things*, vol. 9, Mar. 2020, Art. no. 100164.
[131] L. Sciullo, F. Montori, A. Trotta, M. Di Felice, and T. S. Cinotti, "Discovering web things as services within the arrowhead framework," in *Proc. IEEE Conf. Ind. Cyberphys. Syst. (ICPS)*, Jun. 2020, pp. 571–576.
[132] L. Sciullo, A. Trotta, L. Gigli, and M. D. Felice, "Deploying W3C web of things-based interoperable mash-up applications for industry 4.0: A testbed," in *Proc. Int. Conf. Wired/Wireless Internet Commun.* Cham, Switzerland: Springer, 2019, pp. 3–14.
[133] L. Sciullo, I. D. R. Zyrianoff, A. Trotta, and M. Felice, "WoT micro servient: Bringing the W3C web ofthings to resource constrained edge devices," in *Proc. 7th Int. Conf. Smart Comput. (SMARTCOMP)*, 2021, pp. 161–168.
[134] M. Serrano, P. Barnaghi, F. Carrez, P. Cousin, O. Vermesan, and P. Friess, "Internet of Things IoT semantic interoperability: Research challenges, best practices, recommendations and next steps," Eur. Res. Cluster Internet Things, Tech. Rep. IERC-AC4, 2015.
[135] M. Shoaib, W.-C. Song, R. Ahamd, and D.-H. Kim, "Architecture of push service based on SNS for sharing sensor information," in *Green and Smart Technology With Sensor Applications*. Berlin, Germany: Springer, 2011, pp. 342–346.
[136] B. N. Silva, M. Khan, K. Lee, Y. Yoon, D. Muhammad, J. Han, and K. Han, "RESTful web of things for ubiquitous smart home energy management," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2020, pp. 176–180.
[137] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, and I. P. Žarko, "OpenIoT: Open source Internet-of-Things in the cloud," in *Interoperability and Open-Source Solutions for the Internet of Things*. Cham, Switzerland: Springer, 2015, pp. 13–25.
[138] H. Son, B. Kim, T. Kim, D. Lee, and S.-J. Hyun, "Sherlock-SD: A lightweight universal service discovery for web of things (WoT) services," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2015, pp. 276–282.
[139] *SWAMP Eu Project*, 2020.
[140] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 20–26, Jun. 2014.
[141] I. Szilagyi and P. Wira, "Ontologies and semantic web for the Internet of Things—A survey," in *Proc. 42nd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2016, pp. 6949–6954.
[142] L. Tan and N. Wang, "Future internet: The Internet of Things," in *Proc. 3rd Int. Conf. Adv. Comput. Theory Eng. (ICACTE)*, vol. 5, 2010, p. V5-376.
[143] M. Tang, Y. Xia, B. Tang, Y. Zhou, B. Cao, and R. Hu, "Mining collaboration patterns between APIs for mashup creation in web of Things," *IEEE Access*, vol. 7, pp. 14206–14215, 2019.
[144] J. G. Terius-Padron, E. Simeoni, R. I. Garcia-Betances, N. Liappas, E. Gaeta, M. F. Cabrera-Umpierrez, and M. T. A. Waldmeyer, "Autonomus air quality management system based on web of things standard architecture," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Internet People Smart City Innov., SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019*, Aug. 2019, pp. 184–189.
[145] A. S. Thuluva, D. Anicic, and S. Rudolph, "Semantic web of things for industry 4.0," in *Proc. RuleML+RR*, 2017, p. 883.
[146] L. Tobarra, A. Robles-Gómez, R. Pastor, R. Hernández, J. Cano, and D. López, "Web of things platforms for distance learning scenarios in computer science disciplines: A practical approach," *Technologies*, vol. 7, no. 1, p. 17, Jan. 2019.
[147] V. Trifa, D. Guinard, and D. Carrera. (2015). *Web Thing Model*. [Online]. Available: https://www.w3.org/Submission/wot-model/
[148] V. Trifa, S. Wieland, D. Guinard, and T. M. Bohnert, "Design and implementation of a gateway for web-based interaction and management of embedded devices," in *Proc. 2nd Int. Workshop Sensor Netw. Eng. (IWSNE)*, 2009, pp. 1–14. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.155.4806&rep=rep1&type=pdf
[149] H. D. Veer and A. Wiles, "Achieving technical interoperability: The ETSI approach," 3rd ed., Eur. Telecommun. Standards Inst., Sophia Antipolis Cedex, France, ETSI White Paper 3, Apr. 2008, p. 29. [Online]. Available: https://portal.etsi.org/CTI/Downloads/ETSIApproach/IOPwhitepaper Edition 3 final.pdf
[150] D. Ventura, R. Verborgh, V. Catania, and E. Mannens, "Autonomous composition and execution of REST APIs for smart sensors," in *Proc. Joint 1st Joint Int. Workshop Semantic Sensor Netw. Terra Cognita 4th Int. Workshop Ordering Reasoning*, vol. 1488, 2015, pp. 25–30.
[151] R. Verborgh, T. Steiner, D. V. Deursen, J. D. Roo, R. V. D. Walle, and A. J. GabarróVallés, "Capturing the functionality of web services with functional descriptions," *Multimedia Tools Appl.*, vol. 64, no. 2, pp. 365–387, 2013, doi: 10.1007/s11042-012-1004-5.
[152] W3C. (2017). *Semantic Sensor Network Ontology*. [Online]. Available: https://www.w3.org/TR/vocab-ssn/
[153] W3C. (2021). *Eclipse Thingweb Node-WoT*. [Online]. Available: https://github.com/eclipse/thingweb.node-wot
[154] W3C Working Group. (2019). *Web of Things (WoT) Security and Privacy Guidelines (Working Group Note 6 November 2019)*. [Online]. Available: https://www.w3.org/TR/wot-security/
[155] W3C Working Group. (2020). *WoT Reference Architecture (Proposed Recommendation 9 April 2020)*. [Online]. Available: http://www.w3.org/TR/wot-architecture/
[156] W3C Working Group. (2021). *WoT Reference Architecture (Working Draft 2 June 2021)*. [Online]. Available: https://www.w3.org/TR/wot-discovery/
[157] E. Wilde, "Putting things to REST," *Transport*, vol. 15, pp. 1–13, Nov. 2007. [Online]. Available: http://dret.net/netdret/publications#wil07n

[158] World Wide Web Consortium (W3C). (2018). *WoT Interworking. oneM2M*. [Online]. Available: https://www.onem2m.org/component/rsfiles/download-file/files?path=Draft_TR%5CTR-0042-WoT_Interworking-V0_4_0.doc&Itemid=238
[159] D. Zeng, S. Guo, and Z. Cheng, "The web of things: A survey," *J. Commun.*, vol. 6, no. 6, pp. 424–438, 2011.
[160] C. Zhang, C. Cheng, and Y. Ji, "Architecture design for social web of things," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1–7.
[161] Y. Zhou, S. De, W. Wang, and K. Moessner, "Search techniques for the web of things: A taxonomy and survey," *Sensors*, vol. 16, no. 5, p. 600, Apr. 2016, doi: 10.3390/s16050600.
[162] F. Zonzini, C. Aguzzi, L. Gigli, L. Sciullo, N. Testoni, L. De Marchi, M. Di Felice, T. S. Cinotti, C. Mennuti, and A. Marzani, "Structural health monitoring and prognostic of industrial plants and civil structures: A sensor to cloud architecture," *IEEE Instrum. Meas. Mag.*, vol. 23, no. 9, pp. 21–27, Dec. 2020.
[163] I. Zyrianoff, A. Heideker, L. Sciullo, C. Kamienski, and M. Di Felice, "Interoperability in open IoT platforms: WoT-FIWARE comparison and integration," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Aug. 2021, pp. 169–174.

LUCA SCIULLO received the master's degree *(summa cum laude)* in computer science and the Ph.D. degree in computer science and engineering from the University of Bologna, Italy, in 2017 and 2021, respectively. He was a Visiting Researcher at the Huawei European Research Center of Munich, Germany. He is a Postdoctoral Researcher with the University of Bologna. He is part of the IoT Prism Laboratory directed by Prof. Marco Di Felice and Prof. Luciano Bononi. His research interests include wireless systems and protocols for emergency scenarios, wireless sensor networks, the IoT systems and the web of things.

LORENZO GIGLI received the master's degree *(summa cum laude)* in computer science from the University of Bologna, Italy, in 2019, where he is currently pursuing the Ph.D. degree in engineering and information technology. He was a Research Fellow on the MAC4PRO Project (INAIL) at the Department of Computer Science and Engineering (DISI), University of Bologna. He is part of the IoT PRISM laboratory directed by Prof. Marco Di Felice and the Research and Development Departments of Epoca S.r.l. and VAIMEE S.r.l. His research interests include the IoT, the WoT, distributed systems, containers, and cloud architectures.

FEDERICO MONTORI (Member, IEEE) received the B.S. and M.S. degrees *(summa cum laude)* in computer science and the Ph.D. degree in computer science and engineering from the University of Bologna, Italy, in 2012, 2015, and 2019, respectively. He was a Visiting Researcher with the Swinburne University of Technology, Australia, and Luleå Tekniska Universitet, Sweden. He is currently an Assistant Professor with the University of Bologna, since February 2022. He participated in several EU projects and he is currently WP Leader for the H2020 Project Arrowhead Tools. His primary research interests include mobile crowdsensing (MCS), pervasive and mobile computing, the IoT automation, and data analysis for the IoT scenarios.

ANGELO TROTTA (Member, IEEE) received the PhD. degree in computer science and engineering from the University of Bologna, Bologna, Italy, in 2017. He was a Visiting Researcher at the Heudiasyc Laboratory, Sorbonne Universities, UTC, Compiegne, France, and with the Genesys-Laboratory, Northeastern University, Boston, MA, USA. He is a Junior Assistant Professor with the Department of Computer Science and Engineering, University of Bologna. He is Co-Founder of AI for People—an international association whose aim is to use the artificial intelligent technology for the social good. His current research interests include nature inspired algorithms for self-organizing multirobots wireless systems, reinforcement learning, 5G slicing, and the IoT.

MARCO DI FELICE (Member, IEEE) received the Laurea *(summa cum laude)* and Ph.D. degrees in computer science from the University of Bologna, in 2004 and 2008, respectively. He was a Visiting Researcher with the Georgia Institute of Technology, Atlanta, GA, USA and with Northeastern University, Boston, MA, USA. He is a Full Professor of computer science with the University of Bologna, where he is the Co-Director of the IoT PRISM Laboratory. He has authored more than 120 papers on wireless and mobile systems. His research interests include self-organizing wireless networks, unmanned aerial systems, the IoT, the WoT, and the context-aware computing. He was a recipient of the three Best Paper Awards for his scientific production. He is an Associate Editor of the IEEE Internet of Things Journal.