information

MDPI

Article

Models for Internet of Things Environments—A Survey

Ana Cristina Franco da Silva [1,*,†] and Pascal Hirmer [2,†]

1 Institute of Software Engineering, Universität Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
2 Institute for Parallel and Distributed Systems, Universität Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany; pascal.hirmer@ipvs.uni-stuttgart.de
* Correspondence: ana-cristina.franco-da-silva@iste.uni-stuttgart.de
† These authors contributed equally to this work.

Received: 1 September 2020; Accepted: 15 October 2020; Published: 20 October 2020

Abstract: Today, the Internet of Things (IoT) is an emerging topic in research and industry. Famous examples of IoT applications are smart homes, smart cities, and smart factories. Through highly interconnected devices, equipped with sensors and actuators, context-aware approaches can be developed to enable, e.g., monitoring and self-organization. To achieve context-awareness, a large amount of environment models have been developed for the IoT that contain information about the devices of an environment, their attached sensors and actuators, as well as their interconnection. However, these models highly differ in their content, the format being used, for example ontologies or relational models, and the domain to which they are applied. In this article, we present a comparative survey of models for IoT environments. By doing so, we describe and compare the selected models based on a deep literature research. The result is a comparative overview of existing state-of-the-art IoT environment models.

Keywords: Internet of Things; IoT environment model; smart environment

1. Introduction

In recent years, the Internet of Things (IoT) has emerged through inexpensive hardware and their increasing interconnectivity [1]. Hardware devices, equipped with sensors and actuators, can provide access to important context information of their environment to achieve new approaches, such as smart homes or smart cities. In general, an IoT environment can consist of physical hardware components, an IoT platform that connects the hardware to the digital world, and IoT applications that interact with the physical hardware components through the IoT platform [2].

Many IoT environment models have been developed [3], which can describe different aspects of an IoT environment. We divide these aspects into two main layers, as shown in Figure 1, the physical layer and the digital layer. The *physical layer* of an IoT environment refers to aspects describing hardware IoT objects (e.g., devices, sensors, and actuators) and their interconnections. In the *digital layer*, the digital twin is a program that either mirrors a physical device or simulates it [4,5]. Hence, it refers to aspects describing running services provided within an IoT environment. The IoT application logic refers to models that logically use the services provided by the digital twin to achieve the specific goals of an IoT application, e.g., situation recognition or dashboards.

Currently, IoT environment models highly differ in their content, formats, and the domain to which they are applied. Examples for such models are SensorML [6] or IoT-Lite [7]. Some of these models are maintained by large organizations; others have been created in research projects and are maintained by a small group of people. Furthermore, some of them are even standardized. In order to support IoT application developers in finding a suitable model for their use cases, as well as researchers

in getting an overview of existing IoT environment models, in this article, we present a comparative survey of several models to describe IoT environments. By doing so, we consider not only standards, but also new approaches that are not yet fully mature.

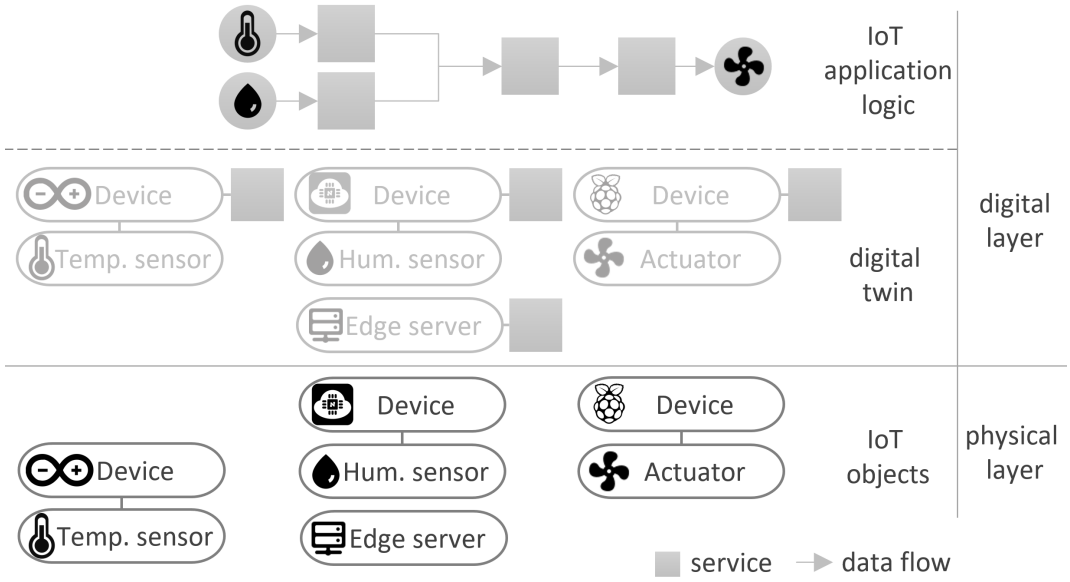

Figure 1. Layers of the Internet of Things.

The remainder of this article is structured as follows: In Section 2, we provide necessary foundational background concepts for this survey. Section 3 gives an overview of IoT environment models that are considered in this article. Section 4 describes criteria to which IoT environment models are evaluated against and compares these models. Section 5 describes related work, and finally, Section 6 gives a summary of this article.

2. Background

In this section, we describe the foundational background for our survey, i.e., the Internet of Things, IoT environment models, and ontology models.

2.1. Internet of Things and IoT Environment Models

In the Internet of Things (IoT), multiple devices communicate with each other through uniform network addressing schemes to reach common goals [8]. These devices are usually equipped with sensors and actuators to monitor environments and act upon situations. The communication among these devices, however, is very complex and heterogeneous. This complexity originates from different communication protocols, data formats, and interfaces. Furthermore, the large amount of devices existing in an IoT environment omits a clear overview, for example, for IoT application developers.

We define an IoT environment model based on previous work [9] as follows: an IoT environment model contains representations of (i) devices, sensors, and actuators of the IoT environment and (ii) the connections among them. In order to define which devices, sensors, actuators, communication protocols, or data formats are contained within an IoT environment, a large amount of models were developed that aim at a standardized description of such complex environments. However, these models differ greatly regarding the abstraction level, focus, or genericity. For example, some models focus on the network; others focus on the physical characteristics of the devices. In addition, some models describe the characteristics of the environment itself (e.g., a factory); others do not.

For IoT application developers, these models can be very helpful to understand the structure of the IoT environment or even be used as an underlying data model for their applications. However,

choosing the right model for a specific application is a difficult task. The wrong decision could lead to a *lock-in*, because changing the underlying data model of an application is error-prone and time-consuming. In this article, we present a survey of IoT environment models in order to support IoT application developers to choose a suitable model for their applications.

2.2. Ontology Models

Ontologies are an important concept in the scope of IoT environment models and, thus, also for this article. Ontologies offer a possibility to describe semantics between entities, leading to large, semantic graphs. Each entity usually has relations of the type *subject (S)*, *predicate (P)*, and *object (O)*. Through these relations, semantics can be expressed [10]. For example, the IoT device *Raspberry Pi* (the subject) is *located* (the predicate) in *room B* (the object). Furthermore, the subject *Raspberry Pi* (S) could be *connected* (P) to an *Arduino board* (O). A famous language to create ontologies is the Web Ontology Language 2 (OWL 2).

Especially for the IoT, using ontology models makes sense because a device is highly interconnected with other devices and with its surroundings. For example, a device can be located in a room of a building, is attached with sensors and actuators, and has connections to other devices. Using the already in place graph model of ontologies enables a means to describe such dependencies in an easy manner. Consequently, many of the developed models mentioned in this article are based on ontology models to describe IoT environments.

Figure 2 shows an example ontology schema for the IoT. In this model, sensors and devices are entities derived from a generic *Object* entity. Each device contains sensors, and each sensor is connected to an adapter, which provides access to it. Actuators are omitted in this simple example. This example shows how ontology models can be used in order to describe IoT environments. Many models introduced in this survey are similar to this example.

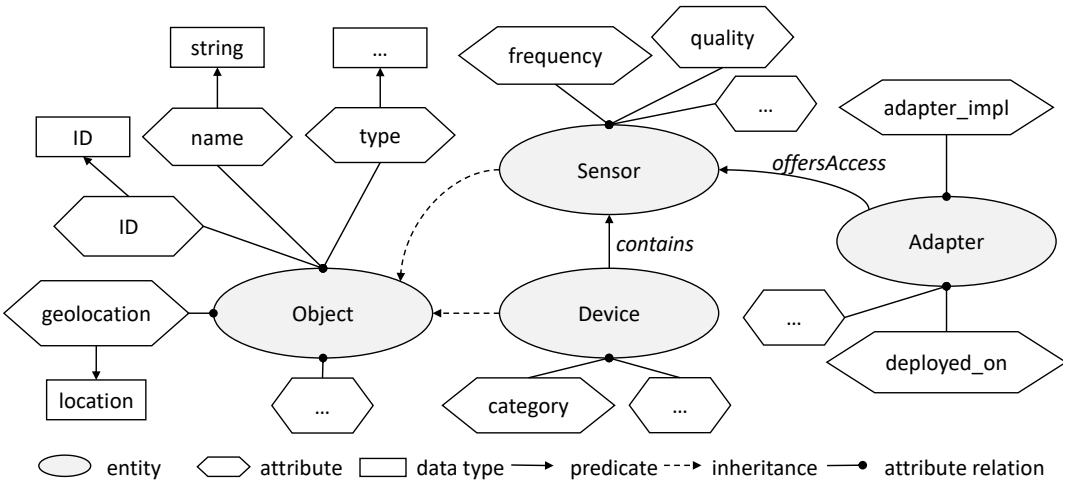

Figure 2. Example of an ontology-based model for IoT environments based on [11].

3. Models for IoT Environments

In this section, we survey several models that can be used to describe the components of IoT environments, e.g., devices, sensors, and actuators.

Our research methodology is depicted in Figure 3, which is comprised of three main steps. In order to compare IoT environment models, we first define five criteria covering important characteristics of IoT models. These criteria were identified from experience and interviews in the scope of the German industry projects SmartOrchestra [12] and Industrial Communication for Factories (IC4F) [13]. These projects have many industry partners with expertise in the IoT and Industry 4.0 domains. Afterwards, we searched for models in two different databases by using combinations of the following keywords:

"*IoT*", "*Internet of Things*", "*environment model*", and "*ontology*". The following search string was used: "*((*"*Internet of Things environment model*" *OR* "*IoT environment model*"*) OR (*"*Internet of Things ontology*" *OR* "*IoT ontology*"*))*". Papers that were found by the search string were selected or excluded based on their title, abstract, and full-text reading. We excluded papers without accessible full-texts in the selected databases, not written in English, or that were not peer reviewed. Papers that did not have the focus on modeling any aspect of IoT environments based on their paper title or abstract were also excluded. Furthermore, models that showed dependencies on specific technologies (e.g., ZigBee) or systems (e.g., IoT platforms) were excluded as well. Moreover, we also took into consideration models that were referred to by industry partners. In the last step, the resulting fifteen models were compared based on the defined criteria. The criteria and criteria-based comparison are presented in Section 4. The publication period of the compared models ranges from 1998 to 2019.

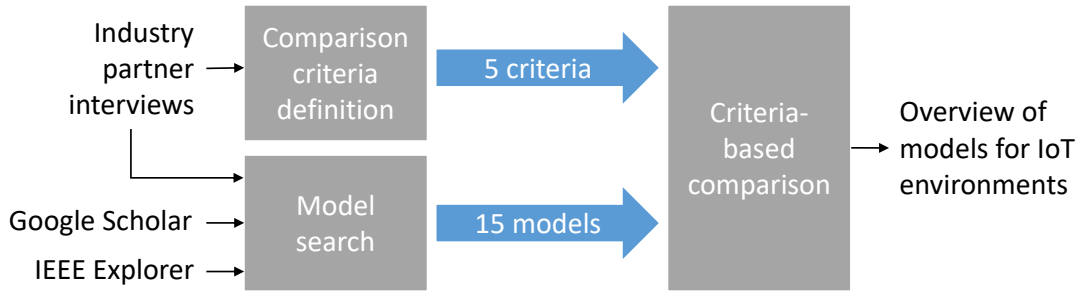

Figure 3. Research methodology.

The selected IoT environment models are described in the following. Note that this article does not aim to describe these models in detail, but rather to enable a broad overview and comparison in order to support the choice of a suitable model for IoT application developers. The following models are ordered alphabetically.

homeML is an XML-based open format for the data exchange within smart home environments proposed by Nugent et al. [14]. This format enables the description of a smart home, its rooms, and sensors within the rooms. Actuators and network communication are not considered by this model. Consequently, homeML focuses on the physical layer, e.g., the devices (cf. Figure 1). A device description in homeML contains, for example, the device ID, location, and type, as well as a description of events generated by the device. In the following, an example of homeML based on [14,15] is depicted.

```
<?xml version="1.0" encoding="UTF-8"?>
<homeML>
 <inhabitantDetails>
  <inhabitantID>454542</inhabitantID>
 </inhabitantDetails>
 <location>
  <locationID>4754</locationID>
  <locationDescription>Living Room</locationDescription>
  <locationDevice>
   <ldeviceID>454584</ldeviceID>
   <deviceType>Temp Sensor</deviceType>
   <units>Celsius<units>
   <deviceDescription>Temperature Sensor Living Room</deviceDescription>
   <deviceLocation>
    <xPos>10.5</xPos>
    <yPos>12.3</yPos>
    <zPos>54.2</zPos>
   </deviceLocation>
```

```
    <event>
     <eventID>1</eventID>
     <timeStamp>11:20:50, 9/24/11</timeStamp>
     <data>23.1</data>
    </event>
    <event>
     <eventID>2</eventID>
     <timeStamp>11:21:44, 9/24/11</timeStamp>
     <data>23.2</data>
    </event>
   </locationDevice>
  </location>
  <annotationDetails>
   <annotationID>4654654</annotationID>
   <IDevice>454584</IDevice>
   <startTimeStamp>11:20:50, 9/24/11</startTimeStamp>
   <endTimeStamp>11:21:44, 9/24/11</endTimeStamp>
  </annotationDetails>
</homeML>
```

In order to support the use of homeML, the modeling tool homeML suite was introduced by McDonald et al. [15]. A new and extended version of homeML (Version 2.2) was introduced in 2013. homeML is specialized for smart homes and does not aim at being a generic model for the IoT.

IEEE 1451.2 is part of a group of IEEE standards aiming to ease sensor and actuator connectivity [16,17]. It specifies the transducer electronic data sheet (TEDS), which allows the self-description of transducers, i.e., sensors or actuators. Such a description is stored in a nonvolatile memory and contains, for example, the type, operation, and calibration of a sensor. In Table 1, an example of a TEDS for a voltage sensor provided by the manufacturer Futek [18] is shown.

This standard has been adapted to the IEEE standard 21450 [19]. Similar to homeML, these standards focus on the physical layer, especially on sensors and actuators. IEEE 1451.2 and 21450 aim to provide generic models for the IoT and are very comprehensive. As an IEEE standard, these models are reviewed and approved by an expert committee. Hence, these standards can be adopted by companies in order to realize complex IoT applications. Implementations of this standard have been developed by Conway et al. [20] and by Song and Lee [21]. Furthermore, Cherian et al. [22] employed this standard to connect industrial legacy sensors to Ethernet networks.

The **IoT-Lite** ontology [7,23] has been developed within the European funded projects FIWARE (https://www.fiware.org) and FIESTA-IoT (http://fiesta-iot.eu) and is a lightweight instantiation of the Semantic Sensor Network (SSN) ontology to represent resources, entities, and services in heterogeneous IoT platforms. In IoT-Lite (cf. Figure 4), IoT devices are sub-classified as sensing devices, actuating devices, and tag devices, e.g., a radio-frequency identification (RFID) tag. Moreover, IoT-Lite defines (i) *objects*, an IoT entity, (ii) *system or resource*, an abstraction for sensing infrastructure, and (iii) *services* provided by IoT devices. Currently, IoT-Lite is under submission as a standard at the World Wide Web Consortium (W3C) organization, where several implementation examples of the ontology are provided.

Table 1. Exemplary transducer electronic data sheet (TEDS) of a voltage sensor (based on [18]).

TEDS Structure	Property	Value	Units
Basic TEDS (64 bits)	Manufacturer ID	Futek Advanced Sensor Technology, Inc.	-
	Model number	MP	-
	Version letter	P	-
	Version number	300	-
	Serial number	123456	-
TEDS template: High-level Voltage output (154 to 253 bits)	Template ID	30	-
	Physical Measurand (Units)	psi	-
	Minimum physical value	0	psi
	Maximum physical value	50	psi
	Transducer electrical signal type	Voltage sensor	-
	Full-scale electrical value precision	0-10V	-
	Minimum voltage output	0	V
	Maximum voltage output	10	V
	Mapping method	Linear	-
	AC or DC coupling	DC	-
	Sensor output impedance	1	Ohms
	Response time	0.001	s
	Excitation/power requirements	Power supply/excitation source	-
	Power supply level, nominal	24	V
	Power supply level, minimum	14	V
	Power supply level, maximum	30	V
	Power supply type	DC	-
	Maximum current at nominal power level	0.001	A
	Calibration date	11/3/2016	-
	Calibration initials	NWH	-
	Calibration period	365	days
	Measurement location ID	1	-
User data	-	-	

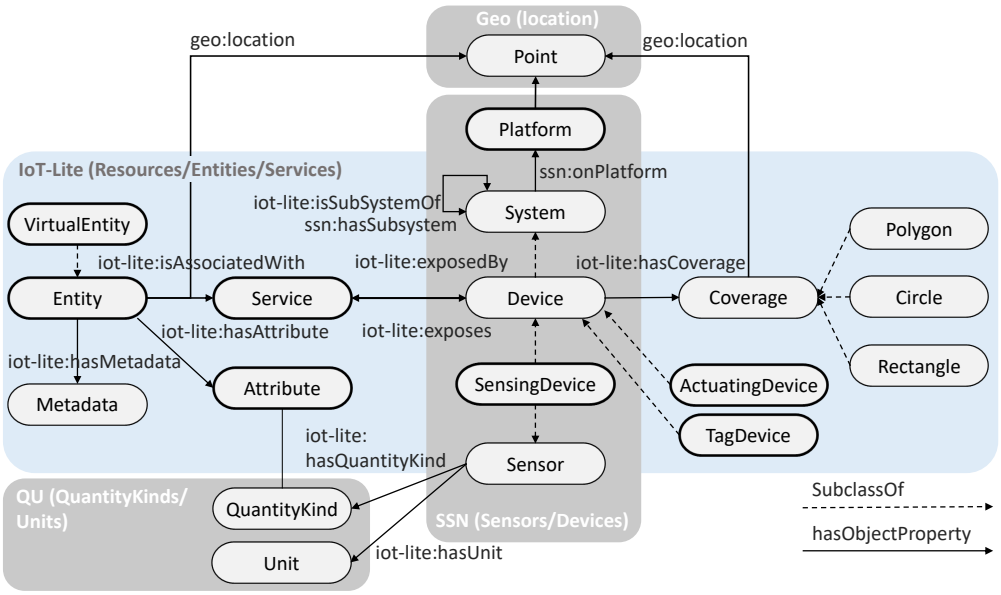

Figure 4. IoT-Lite ontology (based on [23]).

IoT-Lite is generic and aims at reducing the complexity of other IoT models by describing only the main IoT concepts; however, it can be extended to represent IoT concepts in detail and in different

domains. In conclusion, IoT-Lite enables describing IoT environments throughout all the IoT layers depicted in Figure 1.

The **IoT-Stream** ontology [24,25] is a lightweight model to semantically annotate IoT data streams generated within IoT environments. This model corresponds to an extension of the SSN ontology and its lightweight core Sensor, Observation, Sample, and Actuator (SOSA) [26]. Their information model focuses on modeling stream observations and their analysis and events that are detected from it, which are captured in four classes: *IotStream*, *StreamObservation*, *Analyticsprocess*, and *Event* classes. IoT-Stream includes modules for annotation, consumption, and querying of data. Furthermore, tools are included that facilitate the use of semantics in IoT. IoT-Stream focuses on the IoT application logic layer, as depicted in Figure 1.

The **IoT Architectural Reference Model (IoT ARM)** [27] was developed in the scope of the European Lighthouse project IoT-A (https://cordis.europa.eu/project/id/257521) and consists of several sub-models that address architecture views for the IoT. Its primary model is the IoT Domain Model, which describes the main concepts of the IoT (e.g., devices, services, virtual entities) and the relations among theses concepts. Furthermore, this model supports the modeling of users and their interaction with physical entities in the physical world. A physical entity is represented in the digital world by a virtual counterpart called the virtual entity. Interactions with a physical entity are realized through devices, which provide interfaces, for example to gain information about sensors or to control actuators. In conclusion, IoT ARM focuses on the physical layer and the digital twin depicted in Figure 1.

The Open Connectivity Foundation (OCF) **IoT Management and Control** specification enables standardized device and service descriptions for the management and control of IoT environments using Universal Plug and Play (UPnP) technology [28]. One of its main goals is to describe sensors and actuators of UPnP and non-UPnP networks as well; thus, it considers network characteristics, as well as physical characteristics. It also provides a means to define general-purpose devices that are connected to sensors and actuators. A reference implementation of the OCF specification has been developed in the IoTivity project [29]. OCF IoT Management and Control is generic and does not focus on a specific IoT domain; hence, it can be applied to different use cases and applications. Furthermore, it describes the physical layer, as well as the digital twin layer depicted in Figure 1.

IoT-O is an ontology proposed by Alaya et al. [30], which extends the oneM2M standard to support semantic data interoperability. IoT-O handles the sensing, actuating, and service concepts of M2M devices. In order to cover all relevant IoT concepts, it comprises a set of existing ontologies, i.e., DUL (Dolce-UltraLite), SSN, SAN, QUDV, OWL-TIME, and MSM ontologies, which were selected to describe five main concepts: sensor, observation, actuator, actuation, and services models. Consequently, IoT-O is a very comprehensive ontology combining different existing approaches; however, it is currently not a standard, and no reference implementation could be found. In conclusion, IoT-O is generic and enables a wide range of use case scenarios in different IoT domains. IoT-O also enables description throughout all layers of the IoT (cf. Figure 1).

The **Nexus Augmented World Model (AWM)** is part of the *Nexus Platform*, a framework for context-aware applications [31]. Nexus AWM is an abstract model for temporal-spatial context data, which is composed of tuples containing location, time, and type attributes. Furthermore, the abstract model can be extended to be employed in specific domains, such as a smart factory or a smart home. Nicklas et al. [32] proposed the Nexus metadata model, which enables the representation of physical and logical entities. This metadata model serves as a foundation for the Nexus AWM, as well as an integration schema to achieve global integration architectures for mobile, context-aware applications. Similar to IoT-O and other models, Nexus AWM enables description throughout all layers of the IoT (cf. Figure 1).

The **oneM2M base ontology** [33] is an IoT standard that specifies the semantics of the data handled in the oneM2M specification. This ontology defines a device as a derivation of a generic *thing* designed to accomplish a particular task through functions of the device. These functions are exposed

in the network as services of the device. In the context of oneM2M, a device is assumed to be always capable of communicating via a network. Furthermore, sensors and actuators are abstracted as devices, whose concrete functionalities can be expressed as services. Thus, sensors and actuators are implicitly represented via their services. In conclusion, the oneM2M base ontology describes the physical and digital twin layers.

The **OPC UA Information Model** (IEC 62541-5) [34] belongs to the OPC Unified Architecture (OPC UA) standard, which focuses on the interoperable, secure, and reliable exchange of data in the scope of industrial communication. It guarantees, therefore, platform independence and seamless flows of information among devices from multiple vendors. OPC UA defines a client-server communication paradigm, in which an OPC UA server provides access to data and functions structured as defined by the OPC UA Information Model. OPC UA clients can then interact with the information model through standardized services [35]. An open-source implementation of OPC UA in the programming language C is *open62541* (https://open62541.org). OPC UA focuses mostly on the digital twin layer, i.e., the services and interfaces of the IoT devices, but also enables the description of the physical layer, i.e., the IoT devices, sensors, and actuators.

Sensor Measurement Lists (SenML) is an IETF Internet draft specification for media types to represent measurements and device parameters [36]. SenML provides a common data model to describe measurements and simple metadata about measurements and devices, which can be represented in JavaScript Object Notation (JSON), Concise Binary Object Representation (CBOR), eXtensible Markup Language (XML), and Efficient XML Interchange (EXI). In this model, the data are structured as a single array containing so-called *SenML records*. Each record contains fields, such as the sensor's unique identifier, the measurement time, value, and unit. An example of a temperature sensor measurement in the JSON syntax is shown in the following.

```
[ { "n": "urn:dev:ow:10e2073a01080063",
 "u": "Cel",
 "v": 23.1 } ]
```

In this example, the array has a single record, in which the sensor measures the value 23.1 degrees Celsius. However, SenML does not provide a model to describe an entire IoT environment with its resources, connections, and services. Hence, SenML exclusively focuses on the description of the physical layer. SenML is a standard (RFC 8428 [37]) of the Internet Engineering Steering Group (IESG) and is primarily prominent in scientific publications, such as by Su et al. [38] or by Kaivonen and Ngai [39]. A reference implementation of SenML is provided on the open-source platform GitHub (https://github.com/core-wg/senml-spec).

The **Sensor Model Language (SensorML)** [6] is an Open Geospatial Consortium (OGC) implementation standard, which provides models and XML schema encoding to define processes and processing components involved in measurements and the transformation of observations. Its focus lies on the process of measurement and observation; however, it also provides a means to define the physical characteristics and capabilities of sensors and actuators. Components, such as sensors and actuators, are modeled as physical processes, which can accept one or more inputs and produce one or more outputs. SensorML also supports the linking between processes and, thus, the concept of process chains or workflows. Hence, SensorML is able to describe all layers of the IoT (cf. Figure 1). SensorML is generic; however, the focus lies on physical processes, interconnecting sensed data with actions. Besides the XML representation, an ontology model for SensorML is provided online (http://www.sensorml.com/ontologies.html). In the following, a minimalist example of how a sensor can be described in SensorML is provided.

```
<sml:PhysicalComponent gml:id="temperature_sensor" ... >
 <gml:description>Temperature sensor</gml:description>
 <gml:identifier codeSpace="uid">1</gml:identifier>

 <!-- Observed Property = Output -->
 <sml:outputs>
 <sml:OutputList>
 <sml:output name="temp">
 <swe:Quantity definition=
  "http://sweet.jpl.nasa.gov/2.2/quanTemperature.owl#Temperature">
 <swe:label>Air Temperature</swe:label>
 <swe:uom code="Cel"/>
 </swe:Quantity>
 </sml:output>
 </sml:OutputList>
 </sml:outputs>

 <!-- Sensor Location -->
 <sml:position>
 <gml:Point gml:id="stationLocation"
 srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
 <gml:coordinates>47.8 88.56</gml:coordinates>
 </gml:Point>
 </sml:position>
</sml:PhysicalComponent>
```

The measurements of a sensor are described by the element *sml:outputs*, while the element *sml:position* corresponds to the location of the sensor.

The **SSN ontology** is an OWL 2 ontology to describe sensors and observations, which was developed by the W3C Semantic Sensor Network Incubator group (SSN_XG) [40]. Being a W3C standard, currently, SSN is a widely used ontology, which serves as a basis for many other ontology models, e.g., IoT-Lite [41], IoT-O [30], or the ontology model introduced by Hirmer et al. [11]. The SSN ontology [42] describes sensors with respect to their capabilities, measurement processes, observations, and deployments. It applies the Stimulus-Sensor-Observation (SSO) ontology design pattern to describe the relationships between sensors, stimulus, and observations. SSN provides a good foundation to be extended in order to fit specific use cases. Furthermore, SSN mostly focuses on the physical layer and the digital twin layer. Last year, Janowicz et al. [26] introduced the Sensor, Observation, Sample, and Actuator (SOSA) ontology, which acts as a replacement of SSN's core ontology (SSO). The SOSA ontology is lightweight and general-purpose and models interactions among observations, actuation, and sampling. SOSA resulted in the process of rethinking the SSN ontology based on changes in scope and target audience, technical developments, and lessons learned.

The **TDLIoT** (Topic Description Language for the IoT) is a language for the description of sensors and actuators in the form of *topics*. Topics can be realized through different communication models, such as publish-subscribe or request-response, using different protocols (e.g., HTTP, MQTT, CoAP, or OPC UA). The TDLIoT descriptions define characteristics of the sensors and actuators (physical location, endpoint, hardware type) and of the communication type (protocol, message structure, message format). The following listing shows an example of such a description for a temperature sensor in JSON. In this example, a topic representing a temperature sensor is described with a given location, the structure of the message, the endpoint to access the data, the owner, the protocol used, and the type of topic.

```
{ "data_type": "float",
 "hardware_type": "temperature_sensor",
 "location": {
 "location_type": "city_name",
 "location_value": "Stuttgart"
 },
 "message_format": "JSON",
 "message_structure": {
 "metamodel_type": "JSON_schema",
 "metamodel":"{
  "title": "provider_schema",
  "type": "object",
  "properties": {
   "value": {"type": "float"},
   "timestamp": {"type": "integer"},
   "time_up": {"type": "string"} },
  "required": ["value", "timestamp"]}"
 },
 "middleware_endpoint": "test.mosquitto.org:1883",
 "owner": "city_of_stuttgart",
 "path": "/temperature/celsius",
 "protocol": "MQTT",
 "topic_type": "subscription"
}
```

Currently, the TDLIoT is a research prototype presented by Franco da Silva et al. [43] and focuses on the description of topics. The main goal of the TDLIoT is being extendable in order to fit all IoT related domains. Consequently, concrete parametrization of topic descriptions based on the TDLIoT can be conducted tailored for each specific domain and use case. Hence, the TDLIoT does not focus on a specific domain. In regards to the layers depicted in Figure 1, the TDLIoT focuses on the physical layer, describing sensor and actuator characteristics, and the digital twin layer, describing the data and services provided by the topics. The application logic layer, however, needs to be specified by the IoT applications themselves. The TDLIoT is implemented and open-source available on GitHub (https://github.com/IPVS-AS/TDLIoT).

Eclipse **Vorto** [44] is an IoT open-source development infrastructure for the creation and management of agnostic, abstract device descriptions. A simple language is provided in which devices (e.g., a fitness band) and their functionality (e.g., heart rate monitor and step counter) can be described. These descriptions are then published as information models in a centralized Vorto repository. Since Vorto is provided with a comprehensive programming language, all layers of the IoT can be described. Vorto is an implementation focused approach, which provides several tools to create device descriptions. It aims at a practical approach to describe devices so that they can be directly used for application development. As part of the Eclipse Foundation, Vorto has been developed as open-source and is available on GitHub (https://github.com/eclipse/vorto).

Figure 5 depicts an overview of all aforementioned IoT models and on which aspects of the IoT layers they focus. This figure refers to Figure 1, which shows the different layers of the IoT, the physical layer, including IoT objects, and the digital layer, including the digital twin, as well as the IoT application logic. As shown in Figure 5, most of the investigated models cover the physical layer, as well as the digital twin. Some models, i.e., homeML, IEEE 1451.2, and SenML, only focus on the physical layer. Other models cover all layers, e.g., IoT-O or SensorML, whereas only IoT-Stream focuses on the IoT application logic only. However, since IoT-Stream is an extension of SSN/SOSA,

other layers could be modeled as well. Which model to choose highly depends on the requirements of the specific use case they should be applied and over which layers these requirements spread.

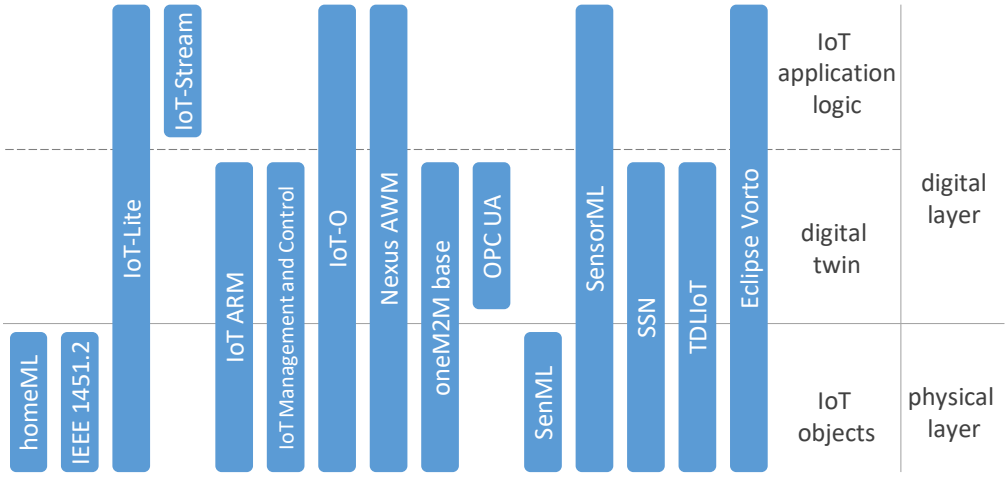

Figure 5. IoT models mapped on layers of the Internet of Things.

4. Criteria-Based Comparison

In order to compare the IoT environment models, we provide five criteria covering their most important characteristics. These criteria are ❶ maturity, ❷ support of hierarchies, ❸ availability and community support, ❹ implementation, and ❺ geolocation support. We identified these criteria by a thorough investigation of available IoT environment models. Furthermore, these criteria originate from experiences in the scope of the German industry projects SmartOrchestra [12] and IC4F [13]. These projects have many industry partners with expertise in IoT applications. These five criteria form the foundation for our survey and are explained in Sections 4.1–4.5. Furthermore, we evaluate the surveyed models based on whether they fulfill the criteria. The criteria-based comparison of the IoT environment models are summarized in Table 2.

Table 2. Criteria-based comparison. ❶: Maturity, ❷: hierarchy, ❸: availability, ❹: implementation, ❺: geolocation. Models marked with * were suggested by industry partners.

Model	❶	❷	❸	❹	❺	Year	Remarks
homeML	non-standard	✗	✗	✗	✓	2007	Designed for smart homes [14,15]
IEEE 1451.2 *	standard	✗	✓	✓	✗	1998	Focuses on sensors [16,17,19]
IoT ARM	non-standard	✓	✗	✗	✓	2013	Generic reference model [27]
IoT-Lite	submitted	✓	✓	✓	✓	2016	Uses SSN ontology [7,23,41]
IoT-Stream	non-standard	✓	✓	✓	✓	2019	Uses SSN ontology [24,25]
IoT MC *	standard	✓	✓	✓	✗	2013	Also known as IoTivity [28,29]
IoT-O	standard ext.	✓	✗	✗	✗	2015	Uses SSN ontology [30]
Nexus AWM *	non-standard	✓	✗	✓	✓	2004	Focuses on geo-localization [31,32]
oneM2M base	standard	✓	✓	✓	✓	2018	Focuses on services of IoT devices [33]
OPC UA *	standard	✓	✓	✓	✗	2016	Established in smart factories [34,35]
SenML *	standard	✗	✓	✓	✓	2012	Focus on sensors and sensor values [36,37]
SensorML	standard	✗	✓	✓	✓	2014	Supports processes [6]
SSN	standard	✓	✓	✓	✓	2005	Well-established IoT ontology [26,42]
TDLIoT *	non-standard	✗	✓	✓	✓	2018	Research prototype [43]
Vorto *	non-standard	✗	✓	✓	✗	2017	Provided as programming language [44]

4.1. Criterion ❶: Maturity (Standard/Non-Standard)

The first criterion specifies the maturity of the models, for example whether a model is an approved standard of an organization, such as OASIS, W3C, or OGC, or not. We assume that a standard that was approved by such organizations went under a thorough reviewing process and, thus, has been checked for feasibility. Furthermore, we assume that a standard has an advantage in contrast to, for example, a work that was published in scientific papers and might have not yet been properly validated or used in real scenarios. Consequently, maturity is an important factor for the evaluation of IoT environment models.

homeML is an academic approach proposed as a collaboration work of the universities of Ulters and Luleå. The *TDLIoT* is also an academic approach presented by the University of Stuttgart. The *IoT-O* ontology is an academic collaboration of the National Center for Scientific Research (CNRS) and the university of Toulose. It is not a standard, but uses the oneM2M standard as a basis and extends it. Furthermore, *IoT ARM* was developed within a European Lighthouse research project, while *Nexus AWM* was the result of a collaborative research center funded by the German research foundation DFG. *IoT-Stream* is not a standard; however, it is very mature, including a detailed documentation and a corresponding implementation.

In contrast to the aforementioned models, several models are approved standards by well-known organizations: *IEEE 1451.2 and 21450* are standards by the IEEE organization; *IoT Management and Control* is a standard by the Open Connectivity Foundation (OCF); *OPC UA* is a standard by the International Electrotechnical Commission (IEC); *SensorML* is an Open Geospatial Consortium (OGC) implementation standard; and the *SSN* ontology is a standard by the W3C Semantic Sensor Network group; furthermore, the *oneM2M base* ontology is a published specification to the oneM2M organization, while *SenML* is a standard by the Internet Engineering Steering Group (IESG). The *IoT-Lite* ontology is currently in submission at the W3C organization.

Finally, *Vorto* is not a standard, but is an open-source tool that has been supported and developed by the Eclipse Foundation.

4.2. Criterion ❷: Support of Hierarchies

Representation of hierarchies is an important factor when modeling environments in the IoT, since they normally contain hierarchical deployments among the different existing IoT objects. There are two main types of hierarchies, *grouping* and *abstraction*. For example, through grouping, it should be possible to model complex systems, such as production machines in a smart factory, which contain a high amount of devices, sensors, and actuators. This enables group-based querying. Such relations can be of vital importance, for example, when conducting monitoring for predictive maintenance [45]. Furthermore, through abstraction, generic types can be defined. For example, different sensor modules measuring temperature can be derived from the generic type *temperature sensor*. Consequently, we investigate whether some support of hierarchies can be expressed in the IoT environment models. For example, an ontology-based model supports natively both mentioned types of hierarchies. Other models normally need to provide such a means separately.

With respect to the surveyed models, many of them do not distinguish among device, sensor, and actuator, i.e., sensors and actuators are abstracted as a device, a thing, or a system, which has specific functionalities for sensing or acting. In this case, we analyze whether hierarchies can be built among devices, things, or systems. If a model provides concepts for grouping, it is indicated in our evaluation.

The first version of *homeML* provides a two level deep hierarchy: A *smart home* must contain at least one *room*, and a room can contain zero or more *devices*. Hierarchies among devices cannot be modeled. Furthermore, devices can be grouped by rooms, i.e., a group of devices existing in a specific room. In the latest version of homeML (Version 2.2), the described hierarchy and grouping are, with minor changes, still kept.

The *IoT-Lite* and *SSN* ontologies define a device as a *system*, which can be a *subsystem* of other systems, enabling in this way hierarchical relations among devices. Similarly, the *IoT-O* ontology defines a device as a *thing*, which can consist of other things, enabling in this way hierarchical relations as well. Since *IoT-Stream* builds on SSN/SOSA, it also supports hierarchical relations. The *oneM2M base* ontology defines a *device*, which can consist of other devices, also enabling hierarchical relations.

The *Nexus Augmented World Model* defines several objects and enables building a hierarchy, in which the root object is called the *NexusObject*. Its children are then *NexusData Objects* containing data of the augmented world. The *OCF IoT Management and Control* standard models hierarchies between devices and sensors. Furthermore, it enables a device to group its sensors into sensor collections. Moreover, sensors of the same type, i.e., temperature sensors, can be modeled as a sensor group, even though they might not belong to the same device.

The *OPC UA Information Model* defines an *AddressSpace*, which contains *Nodes* that can be organized hierarchically or by grouping. The *IoT ARM* abstracts sensors and actuators as devices; however, it is possible to build hierarchies among devices. Furthermore, groups can be modeled as well.

Finally, IEEE 1451.2 describes only one device at once, i.e., no hierarchy or groups can be modeled. Similarly, *SenML*, *SensorML*, *TDLIoT*, and *Vorto* do not support the description of hierarchies.

4.3. Criterion ❸: Availability and Community Support

The third criterion specifies whether the model is publicly available or not and, furthermore, if a wide community is involved in its future development. Clearly, a large community of users and developers, or a larger organization, is required in order to establish and to further develop an IoT environment model. To realize this, the model should either be available open-source, or, if it is closed-source, it should be developed and used by a larger organization.

For *homeML*, a simple link to the XML schema was provided by Nugent et al. [14]; however, the link is not working. No schema could be found for *IoT ARM*; however, there are many UML diagram examples in several related publications [27]. The *IoT-O* ontology is also not available online; the authors only provide a figure of their model.

For the *Nexus Augmented World Model*, the XML schemata could be found in the Nexus project website (https://nexus.uni-stuttgart.de); however, the website was recently deactivated. Regarding the *TDLIoT*, a prototypical implementation is available on GitHub (https://github.com/IPVS-AS/TDLIoT), which provides data storage for topics using the TDLIoT notation and a REST API to publish, update, and search for topics descriptions in this storage.

The *IEEE 1451.2* and *IEEE 21450* standards, *W3C IoT-Lite*, *IoT-Stream*, *OCF IoT Management and Control*, the *oneM2M base* ontology, *OGC SensorML*, and the *SSN_XG SSN* ontology are available online by their corresponding organizations. The *OPC UA Information Model* is available to download upon registration on the OPC Foundation website. Furthermore, the *SenML* specification is available as an Internet draft on the IETF organization website. Finally, *Eclipse Vorto* is available open-source, providing tool support to create information models.

4.4. Criterion ❹: Implementation

The fourth criterion specifies whether an implementation for the IoT environment models exists. In scientific papers, for example, interesting concepts are created that, however, might not have a corresponding implementation. For the usage in real scenarios we aim for in this article, an available implementation is of vital importance. This also includes available tools for model creation and management. For models without a corresponding implementation, we discuss whether an implementation could be realized.

For almost every surveyed model, there are implementations or examples. Only for *homeML*, *IoT ARM*, and *IoT-O*, no implementations were found. A web address to the homeML suite was provided by McDonald et al. [15]; however, the address is not accessible. An implementation of IoT-O

is not available as well. The *Nexus Platform* was implemented; however, this implementation is not available publicly.

An implementation of the *IEEE 1451.2* for sensors with a web address was provided by Wobschall et al. [46]. Furthermore, several implementation examples of the *IoT-Lite* ontology are provided in the submitted document in the W3C organization. *IoT-Stream* provides an implementation of their standard, which is available online (http://iot.ee.surrey.ac.uk/iot-crawler/ontology/iot-stream). A reference implementation of the *OCF IoT Management and Control* specification has been developed in the IoTivity project. An open-source implementation of OPC UA in the programming language C is open62541 (https://open62541.org). Implementations of the *oneM2M base* ontology and *SSN* ontology are provided as OWL files. *SenML* examples for the different formats, as well as an XML schema are provided in the Internet draft. The *SensorML* specification document provides several examples, as well as XML schemata. A prototypical implementation employing the *TDLIoT* is available on GitHub (https://github.com/IPVS-AS/TDLIoT). Finally, several *Vorto* information models for devices, such as Philips Hue and Bosch XDK, are provided.

4.5. Criterion ❺: Geolocation Support

Finally, the fifth criterion defines whether the model can describe the (geo-)location of devices, which enables sophisticated features, such as location-based querying. Especially in the IoT, location is important, for example, when recognizing situations, i.e., events that might require a reaction, which occur in a specific room of a smart home.

Some of the surveyed models do not explicitly provide a concept for the modeling of locations. In this case, we analyze whether the models provide a means to be extended with customized properties in order to enable the definition of the location.

homeML provides the element *DeviceLocation*, which can be used to provide the absolute coordinates (x, y, z) of the device. In *IoT ARM*, a *PhysicalEntity* can contain so-called *Tags*, which can be used to model the location of the physical entity in various formats. The *IoT-Lite* ontology and *SensorML* enable the description of locations. The *SSN* ontology enables the description of locations through the DUL (Dolce-UltraLite) ontology, which provides location concepts. Since *IoT-Stream* builds on SSN/SOSA, it also supports geolocation. In their documentation, they also provide an example of how to model geolocation data using their ontology. The *Nexus Augmented World Model* was designed to support location-aware applications; therefore, it provides a means to model location information.

Furthermore, the *oneM2M base* ontology does not explicitly describe locations; however, this can be modeled as a *variable*, denoting a property of a device, e.g., the location. In the *OPC UA Information Model*, the location can be defined using the so-called *axis*. Each axis is defined by coordinates (x, y, z). *SenML* also does not explicitly describe locations, but its format can be extended with custom attributes.

IEEE 1451.2 allows the self-description of transducers with respect to static, technical properties. Therefore, locations are not described in TEDS. The *IoT Management and Control standard* and the *IoT-O* ontology do not provide a means to describe locations. Finally, the Eclipse *Vorto* information model does not explicitly support location descriptions.

To summarize, currently, there are many application domains for IoT environments. For example, Gubbi et al. [47] provided the following classification of IoT environments based on implemented testbeds: smart home, smart retail, smart city, smart agriculture, smart water, and smart transportation. Moreover, many models to describe IoT environments exist (cf. Section 3). Therefore, application developers have the difficult task of choosing a suitable model for their IoT applications since, typically, the IoT environment model used in such applications cannot be easily exchanged afterwards.

In Table 2, the comparison results based on the introduced criteria are shown. From the surveyed models, homeML is suitable, for example, to describe a smart home, since homeML was explicitly designed for this domain. On the other side, IEEE 1451.2 and OPC UA are rather to be applied in smart factory scenarios. Finally, many models do not define specific domains and are, therefore, generic

enough to be employed in different application domains, such as the models IoT ARM, IoT-Lite, IoT-O, Nexus AWM, oneM2M base, IoT MC, SenML, SensorML, SSN, TDLIoT, and Vorto.

Table 2 shows that there are several models that fulfill all five criteria, i.e., IoT-Lite, IoT-Stream, oneM2M base, and SSN. Note that our comparison only gives a suggestion about which models are suitable for a generic use, i.e., can be used in different application domains. If for example, a smart home application should be developed, specific domain models, such as homeML, might be more suitable, since it was designed for this specific domain and provides additional means to describe home-related concepts (e.g., rooms, floors, inhabitants). However, generic models can also be employed since they normally can be extended or adapted if needed to meet the requirements of a smart home application. Furthermore, a combination of compatible models (e.g., ontologies) is normally possible as well.

In our comparison, we excluded the analysis of the encoding efficiency of the investigated models. Generally, IoT environment models are not created and processed by resource-constrained IoT devices, but rather by more powerful systems, such as IoT platforms hosted on, for example, cloud infrastructures or servers. Hence, we omitted this aspect of the models in this article. However, it is important to note that specifically ontology-based environment models are heavyweight and require a certain amount of main memory, depending on the size of the ontology. In contrast, JSON-based environment models are more lightweight and, thus, easier to create and process. This enables the handling of such environment models by resource-constrained devices as well. Hence, if it is necessary to process the models on resource-constrained devices or transfer them through resource constrained networks, we recommend using lightweight formats, such as SenML, which is able to create compact descriptions of devices that can be used for discovery purposes.

5. Related Work

Compton et al. [48] provided a survey of sensor ontologies. In their work, they focused on the semantic specification of sensors. Their comparison included ontologies such as SWAMO, CSIRO, OntoSensor, and scientific contributions, such as the work introduced by Avancha et al. [49], Matheus et al. [50], or Eid et al. [51]. The criteria for comparison are comprised of (i) sensor specific characteristics, such as sensor hierarchy, identity, manufacturing, contacting, and software, (ii) physical characteristics, such as location, power supply, and operating conditions, (iii) observation specific characteristics, such as accuracy, frequency, and the response model, and (iv) domain specific characteristics, such as units of measurements, feature/quality, or time.

In contrast to Compton et al., our work does not focus on ontologies for the semantic specification of sensors. We compare all kinds of IoT environment models, including semantic and non-semantic models. Furthermore, we do not focus on sensors specifically, but on IoT environments including devices, sensors, and actuators. Moreover, we focus on other criteria, which, in our opinion, are essential to model and describe IoT environments. Finally, most of the models compared by Compton et al. are approximately 10 years old, and newer approaches are therefore missing.

Gyrard et al. [52] created a collection of different vocabularies for the IoT focusing on scientific publications. By doing so, exclusively, ontologies were added to the collection. In contrast, our work focuses not only on ontologies, but on different kinds of models for the IoT.

Chen et al. [53] surveyed several sensor standards, including ECHONET, SensorML, IEEE 1451, Device Kit, and DDL. The authors categorized these standards according to their affiliation to the physical world or the digital world, meaning whether these models describe physical characteristics such as pins or ports or digital characteristics such networking protocols or configuration. In addition, the models were compared regarding the criteria encoding, design perspective, device model, measurement modeling, etc. In our article, we consider non-standard models to enable a wider comparison of the state-of-the-art concepts. Furthermore, this paper was published in the year 2008 and does not consider recent advances in the field of the Internet of Things, including newly emerged models, such as IoT-Lite or the oneM2M base ontology.

Darmois et al. [54] provided a state-of-the-art analysis for IoT standards. Their work was published in the year 2012. Darmois et al. looked at IoT standards in general, not focusing on IoT environment models. They categorized these standards into three layers: (i) the application layer containing high-level IoT applications, (ii) the IoT layer, i.e., the digital representation of the physical world, and (iii) the network layer dealing with the communication. Their overall goal is the recognition of the gaps in the standards of these layers. However, they did not provide a comparison of these standards based on a set of criteria. In contrast, they described existing standards extensively and tried to find gaps that were not addressed by them. Our work does not focus on finding gaps; it aims at providing an overview of existing models to describe IoT environments.

Grangel-González et al. [55] presented a landscape of standards for the Industry 4.0 from a semantic integration perspective. In their work from 2017, they extensively investigated existing standards related to Industry 4.0. These standards include, for example, AML or OPC UA. However, in their work, they did not focus on IoT environment models, but gave an overview of general standards that could be applicable to Industry 4.0. In contrast to their work, the focus of our paper does not lie on Industry 4.0, exclusively. We provide an overview of generic existing IoT environment models, applicable to a wide range of scenarios. In addition, we focus on the digital description of these IoT environments and not, for example, on communication standards.

6. Summary

In this article, we present a comparative survey of models that can be employed to describe IoT environments, including devices, their attached sensors, and actuators. Some of the evaluated models have been developed and maintained by large organizations; others have been created in research projects or have been standardized. This survey supports IoT application developers in finding a suitable model for their use cases, as well as researchers in getting an overview of the state-of-the-art IoT environment models.

In order to compare these different models, we define five criteria that summarize important characteristics of the models to be employed in the IoT domain, such as maturity and available implementations. Based on these criteria, we evaluate and compare the surveyed models. Furthermore, we present related work that analyzes and discusses further models to be used in IoT scenarios.

We are aware that many other models exist that could be relevant for our survey, for example the underlying models used by established IoT products in the smart home domain, such as Amazon Echo, Google Home, or ZigBee Alliance Dotdot [56]. However, to keep the focus clear, we considered some exclusion criteria, as described in Section 3. We did not include the IoT standard Dotdot [56], which is a universal application language provided by the ZigBee Alliance for IoT devices, since it focuses mostly on the device-to-device communication and does not provide a holistic description of IoT environments, including all the involved devices, sensors, and actuators. Furthermore, Amazon Echo and Google Home are also not included, since they do not explicitly provide models to describe them as IoT devices and the corresponding IoT environment of which they are a part.

Author Contributions: Investigation, A.C.F.d.S. and P.H.; methodology, A.C.F.d.S. and P.H.; writing—original draft, A.C.F.d.S. and P.H. All authors read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vermesan, O.; Friess, P. *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*; River Publishers: Aalborg, Denmark, 2013.
2. Franco da Silva, A.C.; Breitenbücher, U.; Hirmer, P.; Képes, K.; Kopp, O.; Leymann, F.; Mitschang, B.; Steinke, R. Internet of Things Out of the Box: Using TOSCA for Automating the Deployment of IoT Environments. In Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER). SciTePress Digital Library, Porto, Portugal, 24–26 April 2017; ScitePress: Setubal, Portugal, 2017; pp. 358–367.
3. Strang, T.; Linnhoff-Popien, C. A Context Modeling Survey. In Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp, Maui, HI, USA, 11–15 September 2004; Volume 4, pp. 34–41.
4. Glaessgen, E.; Stargel, D. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. In Proceedings of the 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Honolulu, HI, USA, 23–26 April 2012.
5. Boschert, S.; Rosen, R. Digital Twin—The Simulation Aspect. In *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers*; Springer International Publishing: New York, NY, USA, 2016; pp. 59–74.
6. OGC. Sensor Model Language (SensorML). 2014. Available online: http://www.opengeospatial.org/standards/sensorml (accessed on 17 October 2020).
7. Bermudez-Edo, M.; Elsaleh, T.; Barnaghi, P.; Taylor, K. IoT-Lite: A Lightweight Semantic Model for the Internet of Things. In Proceedings of the Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences, Toulouse, France, 18–21 July 2016; IEEE: New York, NY, USA, 2016; pp. 90–97.
8. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805.
9. Franco da Silva, A.C.; Hirmer, P.; Mitschang, B. Model-based Operator Placement for Data Processing in IoT Environments. In Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 12–15 June 2019; IEEE: New York, NY, USA, 2019.
10. Guarino, N.; Oberle, D.; Staab, S. What Is an Ontology? In *Handbook on Ontologies*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–17.
11. Hirmer, P.; Wieland, M.; Breitenbücher, U.; Mitschang, B. Automated Sensor Registration, Binding and Sensor Data Provisioning. In Proceedings of the CAiSE'16 Forum, at the 28th International Conference on Advanced Information Systems Engineering, CEUR-WS.org, Ljubljana, Slovenia, 13–17 June 2016; Volume 1612, pp. 81–88.
12. SmartOrchestra Consortium. SmartOrchestra Research Project. 2016. Available online: http://smartorchestra.de/en (accessed on 17 October 2020).
13. IC4F Consortium. IC4F Research Project. 2017. Available online: https://www.ic4f.de (accessed on 17 October 2020).
14. Nugent, C.D.; Finlay, D.D.; Davies, R.J.; Wang, H.Y.; Zheng, H.; Hallberg, J.; Synnes, K.; Mulvenna, M.D. homeML—An Open Standard for the Exchange of Data Within Smart Environments. In *Pervasive Computing for Quality of Life Enhancement: 5th International Conference on Smart Homes and Health Telematics, ICOST 2007, Nara, Japan, June 21–23, 2007. Proceedings*; Chapter Pervasive Computing for Quality of Life Enhancement; Springer: Berlin/Heidelberg, Germany, 2007; pp. 121–129.
15. McDonald, H.; Nugent, C.; Hallberg, J.; Finlay, D.; Moore, G.; Synnes, K. The homeML suite: Shareable datasets for smart home environments. *Health Technol.* **2013**, *3*, 177–193.
16. IEEE. IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats. Standard. 1998. Available online: standards.ieee.org/standard/1451_2-1997.html (accessed on 17 October 2020).
17. Song, E.Y.; Burns, M.; Pandey, A.; Roth, T. IEEE 1451 Smart Sensor Digital Twin Federation for IoT/CPS Research. In Proceedings of the 2019 IEEE Sensors Applications Symposium (SAS), Sophia Antipolis, France, 11–13 March 2019; pp. 1–6.

18. Futek Advanced Sensor Technology, Inc. Transducer Electronic Datasheet—Manual and Programming Guide. Available online: www.futek.com (accessed on 17 October 2020).
19. IEEE. ISO/IEC/IEEE Information technology—Smart transducer interface for sensors and actuators—Common functions, communication protocols, and Transducer Electronic Data Sheet (TEDS) formats. *ISO/IEC/IEEE 21450:2010(E)* **2010**, 1–350.
20. Conway, P.; Heffernan, D.; O'Mara, B.; Burton, P.; Miao, T. IEEE 1451.2: An Interpretation and Example Implementation. In Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference (IMTC), Baltimore, MD, USA, 1–4 May 2000; IEEE: New York, NY, USA, 2000; Volume 2, pp. 535–540.
21. Song, Y.; Lee, K.B. An implementation of the Proposed IEEE 1451.0 and 1451.5 Standards. In Proceedings of the Sensors Applications Symposium, Houston, TX, USA, 7–9 February 2006.
22. Cherian, A.; Wobschall, D.; Sheikholeslami, M. An IoT Interface for Industrial Analog Sensor with IEEE 21451 Protocol. In Proceedings of the IEEE Sensors Applications Symposium (SAS), Glassboro, NJ, USA, 13–15 March 2017; IEEE: New York, NY, USA, 2017; pp. 1–5.
23. Bermudez-Edo, M.; Elsaleh, T.; Barnaghi, P.; Taylor, K. IoT-Lite: A lightweight semantic model for the internet of things and its use with dynamic semantics. *Pers. Ubiquitous Comput.* **2017**, *21*, 475–487.
24. Elsaleh, T.; Bermudez-Edo, M.; Enshaeifar, S.; Acton, S.T.; Rezvani, R.; Barnaghi, P. IoT-Stream: A Lightweight Ontology for Internet of Things Data Streams. In Proceedings of the 2019 Global IoT Summit (GIoTS), Aarhus, Denmark, 17–21 June 2019; pp. 1–6.
25. Elsaleh, T.; Enshaeifar, S.; Rezvani, R.; Acton, S.T.; Janeiko, V.; Bermudez-Edo, M. IoT-Stream: A Lightweight Ontology for Internet of Things Data Streams and Its Use with Data Analytics and Event Detection Services. *Sensors* **2020**, *20*, 953.
26. Janowicz, K.; Haller, A.; Cox, S.J.; Le Phuoc, D.; Lefrançois, M. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *J. Web Semant.* **2019**, *56*, 1–10.
27. Bauer, M.; Bui, N.; De Loof, J.; Magerkurth, C.; Nettsträter, A.; Stefa, J.; Walewski, J.W. IoT Reference Model. In *Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model*; Springer: Berlin/Heidelberg, Germany, 2013; Chapter 7; pp. 113–162.
28. Open Connectivity Foundation. IoT Management and Control. 2013. Available online: https://openconnectivity.org/developer/specifications/upnp-resources/upnp/iot-management-and-control1-2 (accessed on 17 October 2020).
29. Open Connectivity Foundation. IoTivity. 2017. Available online: https://iotivity.org (accessed on 17 October 2020).
30. Alaya, M.B.; Medjiah, S.; Monteil, T.; Drira, K. Toward Semantic Interoperability in oneM2M Architecture. *IEEE Commun. Mag.* **2015**, *53*, 35–41.
31. Dürr, F.; Hönle, N.; Nicklas, D.; Becker, C.; Rothermel, K. *Nexus—A Platform for Context-Aware Applications*; Jörg, R., Ed.; Gesellschaft für Informatik (GI): Hagen, Germany, 24 June 2004; Volume 1, pp. 15–18.
32. Nicklas, D.; Schwarz, T.; Mitschang, B. A Schema-Based Approach to Enable Data Integration on the Fly. *Int. J. Coop. Inf. Syst.* **2017**, *26*, 1650010.
33. oneM2M Partners. oneM2M Base Ontology. 2018. Available online: http://www.onem2m.org/technical/latest-drafts (accessed on 17 October 2020).
34. OPC Foundation. OPC Unified Architecture Specification. Part 5: Information Model, 2017. Standard. Available online: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model (accessed on 17 October 2020).
35. Grüner, S.; Pfrommer, J.; Palm, F. RESTful Industrial Communication With OPC UA. *IEEE Trans. Ind. Inform.* **2016**, *12*, 1832–1841.
36. Jennings, C.; Arkko, J.; Shelby, Z. Media Types for Sensor Markup Language (SENML). 2012. Available online: https://tools.ietf.org/html/draft-jennings-senml-10 (accessed on 17 October 2020).
37. Jennings, C.; Shelby, Z.; Arkko, J.; Keranen, A.; Bormann, C. Sensor Measurement Lists (SenML), 2018, Standard. Available online: https://tools.ietf.org/html/rfc8428 (accessed on 17 October 2020).
38. Su, X.; Zhang, H.; Riekki, J.; Keränen, A.; Nurminen, J.K.; Du, L. Connecting IoT Sensors to Knowledge-based Systems by Transforming SenML to RDF. *Procedia Comput. Sci.* **2014**, *32*, 215–222.
39. Kaivonen, S.; Ngai, E.C.H. Real-time air pollution monitoring with sensors on city bus. *Digit. Commun. Netw.* **2020**, *6*, 23–30.

40. Compton, M.; Barnaghi, P.; Bermudez, L.; García-Castro, R.; Corcho, O.; Cox, S.; Graybeal, J.; Hauswirth, M.; Henson, C.; Herzog, A.; et al. The SSN ontology of the W3C semantic sensor network incubator group. *J. Web Semant.* **2012**, *17*, 25–32.
41. W3C. IoT-Lite Ontology. 2015. Available online: https://www.w3.org/Submission/2015/SUBM-iot-lite-20151126 (accessed on 17 October 2020).
42. W3C. Semantic Sensor Network Ontology. 2005. Available online: https://www.w3.org/2005/Incubator/ssn/ssnx/ssn (accessed on 17 October 2020).
43. Franco da Silva, A.C.; Hirmer, P.; Breitenbücher, U.; Kopp, O.; Mitschang, B. TDLIoT: A Topic Description Language for the Internet of Things. In Proceedings of the International Conference on Web Engineering (ICWE), Cáceres, Spain, 5–8 June 2018; Springer International Publishing: New York, NY, USA, 2018; pp. 333–348.
44. Eclipse. Vorto. 2017. Available online: https://www.eclipse.org/vorto (accessed on 17 October 2020).
45. March, S.T.; Scudder, G.D. Predictive maintenance: Strategic use of IT in manufacturing organizations. *Inf. Syst. Front.* **2017**, *21*, 1–15.
46. Wobschall, D. An implementation of IEEE 1451 NCAP for Internet access of serial port-based sensors. In Proceedings of the 2nd ISA/IEEE Sensors for Industry Conference, Houston, TX, USA, 19–21 November 2002; pp. 157–160.
47. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660.
48. Compton, M.; Henson, C.; Lefort, L.; Neuhaus, H.; Sheth, A. A Survey of the Semantic Specification of Sensors. In Proceedings of the 2nd International Conference on Semantic Sensor Networks (SSN'09), Washington DC, USA, 26 October 2009; Volume 522, pp. 17–32.
49. Avancha, S.; Patel, C.; Joshi, A. Ontology-driven adaptive sensor networks. In Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous), Boston, MA, USA, 22–26 August 2004; pp. 194–202.
50. Matheus, C.J.; Tribble, D.; Kokar, M.M.; Ceruti, M.G.; McGirr, S.C. *Towards a Formal Pedigree Ontology for Level-One Sensor Fusion*; Technical Report; Versatile Information Systems Inc.: Riverside, CA, USA, 2006.
51. Eid, M.; Liscano, R.; el Saddik, A. A Universal Ontology for Sensor Networks Data. In Proceedings of the 2007 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, Ostuni, Italy, 27–29 June 2007; pp. 59–62.
52. Gyrard, A.; Bonnet, C.; Boudaoud, K.; Serrano, M. LOV4IoT: A Second Life for Ontology-Based Domain Knowledge to Build Semantic Web of Things Applications. In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 22–24 August 2016; pp. 254–261.
53. Chen, C.; Helal, S. Sifting Through the Jungle of Sensor Standards. *IEEE Pervasive Comput.* **2008**, *7*, 84–88.
54. Darmois, E.; Elloumi, O.; Guillemin, P.; Moretto, P. IoT Standards–State-of-the-Art Analysis. In *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*; IERC: Cork, Ireland, 2012.
55. Grangel-González, I.; Baptista, P.; Halilaj, L.; Lohmann, S.; Vidal, M.E.; Mader, C.; Auer, S. The Industry 4.0 Standards Landscape from a Semantic Integration Perspective. In Proceedings of the IEEE 22nd International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12–15 September 2017.
56. ZigBee Alliance. Dotdot. 2019. Available online: https://zigbeealliance.org/solution/dotdot (accessed on 17 October 2020).