


Review

Digital Twin and Internet of Things—Current Standards Landscape

Michael Jacoby *  and Thomas Usländer 

Fraunhofer IOSB, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, 76131 Karlsruhe, Germany; thomas.uslaender@iosb.fraunhofer.de

* Correspondence: michael.jacoby@iosb.fraunhofer.de; Tel.: +49-721-6091-470

Received: 20 August 2020; Accepted: 15 September 2020 ; Published: 18 September 2020



Abstract: Industry 4.0 is revolutionizing industrial production by bridging the physical and the virtual worlds and further improving digitalization. Two essential building blocks in industry 4.0 are digital twins (DT) and the internet of things (IoT). While IoT is about connecting resources and collecting data about the physical world, DTs are the virtual representations of resources organizing and managing information and being tightly integrated with artificial intelligence, machine learning and cognitive services to further optimize and automate production. The concepts of DTs and IoT are overlapping when it comes to describing, discovering and accessing resources. Currently, there are multiple DT and IoT standards covering these overlapping aspects created by different organizations with different backgrounds and perspectives. With regard to interoperability, which is presumably the most important aspect of industry 4.0, this barrier needs to be overcome by consolidation of standards. The objective of this paper is to investigate current DT and IoT standards and provide insights to stimulate this consolidation. Overlapping aspects are identified and a classification scheme is created and applied to the standards. The results are compared, aspects with high similarity or divergence are identified and a proposal for stimulating consolidation is presented. Consensus between standards are found regarding the elements a resource should consist of and which serialization format(s) and network protocols to use. Controversial topics include which query language to use for discovery as well as if geo-spatial, temporal and historical data should be explicitly supported.

Keywords: digital twin; internet of things; interoperability; standardization

1. Introduction

The fourth industrial revolution and the underlying digital transformation, known as Industry 4.0, is revolutionizing industrial production. Following the third industrial revolution, also known as the Digital Revolution, Industry 4.0 is about optimization and automation of the previously introduced computerization by intelligent networking of machines and processes [1,2]. A central element is the integration of already-existing concepts such as the internet of things (IoT), digital twins (DT), artificial intelligence (AI), machine learning (ML), big data, and cognitive services and applying them to industrial production. Three of the integral design principles of Industry 4.0 are interconnection, information transparency, and decentralized decisions [3]. Interconnection refers to the ability of machines, devices, sensors, and people to connect and communicate with each other via the internet, which is the cornerstone of the IoT. Information transparency refers to the ability to collect, manage, and organize the vast amount of data from connected machines, devices, and sensors. This requirement is addressed by the concept of DT, which is ‘a formal digital representation of some asset, process or system that captures attributes and behaviors of that entity suitable for communication, storage, interpretation or processing within a certain context’ [4]. The requirement for decentralized decisions can be fulfilled by incorporating aspects of AI and ML into DTs to provide cognitive services and

thereby enable better decision-making and increasing the level of automation as well as overall productivity. DTs with such enhanced capabilities are also referred to as cognitive twins (CT) [5].

The importance of industry 4.0 and its related concepts and technologies has been widely acknowledged [6–9]. Between 2012 and 2016, IoT (or the similar ‘internet of everything’) was considered one of the ten most important technology trends of the year by Gartner, followed by DT as one of the five most important technology trends between 2017 and 2019 [10–17]. In 2020, the most important technology trend according to Gartner is hyperautomation, which refers to the totality of automation efforts across an entire organization and is tightly coupled to industry 4.0 and DT as ‘hyperautomation often results in the creation of a digital twin of the organization’ [18].

It seems obvious that ‘the rise of digital twins coincides with the rise of the IoT’ [19] as only the vast amount of sensor data and device metadata from the IoT creates the requirement for organizing and managing all that information in an adequate way, which is realized by the concept of DTs. Although the relation between DT and IoT seems to be clear at first glance—IoT connects devices to the internet and collects data while the concept of DTs is used to structure and manage that data to be further used for optimization and automation with the help of AI and ML algorithms—they have some overlap and therefore are not clearly separable in some aspects.

A fundamental aspect of both IoT and DT is the fact that they center on resources and require a formal and machine-readable representation of these resources. Resources can be devices, sensors, actuators, or machines in the context of IoT or assets, processes, or systems in the context of DT. Figure 1 provides a visual representation of further resource-related functionality that is part of both domains. Besides a formal and machine-readable resource description created by a provider, they share the functionality for potential consumers to discover such resource descriptions and access the resources.

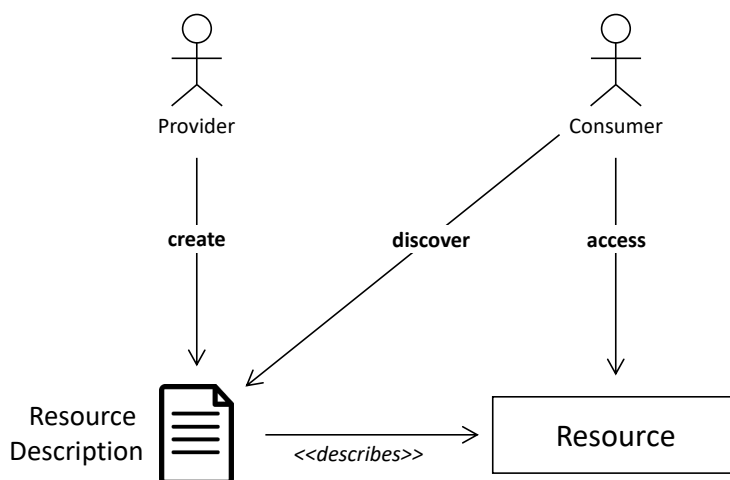


Figure 1. Functional overlap between internet of things and digital twins that is subject to standardization in both domains.

As industry 4.0 is about optimization on a large scale, interoperability not only within a factory or organization but along the whole value chain is essential and should be addressed globally [20,21]. Achieving interoperability on a global level requires formal standardization. Multiple SDOs are currently working on standardization of IoT or DT, e.g., the European Telecommunications Standards Institute (ETSI), the International Electrotechnical Commission (IEC), the Internet Engineering Task Force (IETF), the Industrial Internet Consortium (IIC), the International Organization for Standardization (ISO), the Open Geospatial Consortium, the Plattform Industrie 4.0, the World Wide Web Consortium (W3C), and many more. Many of the standards related to DT or IoT address issues that are unique to the domain, i.e., not part of the overlap between IoT and DT, for example, network protocols for wireless connectivity, which belongs to IoT but is not considered a relevant

subject for standardization in the DT domain. Furthermore, different SDOs have different backgrounds and focuses, e.g., ETSI is rooted in the telecommunication domain, OGC focuses on geospatial information management, and the IIC is focused on accelerating the development and adoption of IoT technology in the industry. Several standards that focus on the three basic aspects common to both IoT and DT, resource description, resource discovery, and resource access, are either already existing or currently being developed by different SDOs in parallel.

Having multiple competing standards solving the same problem in different ways without aligning with each other massively hinders interoperability. As interoperability is the key requirement of industry 4.0 [21,22], this is an actual threat for the further development and success of industry 4.0. Previous work focuses on either the IoT or the DT domain or addresses the IoT only as a source of data for DTs, ignoring the fact that the domains actually overlap in functionality. A recent survey from Minerva et al. acknowledges the fact that interoperability plays an important role in industry 4.0 on IoT on the DT level and that ‘interoperability and standardization will occur as part of the evolution’ [23]. Although the relationship and alignment between existing and upcoming IoT and DT standards have been discussed in multiple SDO groups, to the best of the author’s knowledge, there has not been any published formalized and structured analysis and comparison. The objective of this paper is to narrow this gap and stimulate the evolution of standards by identifying how well current standards from both the IoT and the DT domain are aligned regarding resource description, discovery and access. This not only provides an overview of the current state of the art on DT interoperability, but can also nourish further consolidation of standards to improve interoperability in industry 4.0.

Market research report predicts that until 2027, using DTs will become standard for IoT applications and IoT platforms will support the creation of DTs. At the same time, the total IoT market is predicted to rise from \$465 million in 2019 to \$1.5 trillion in 2030 whereby 40% thereof depend on successfully addressing interoperability [24–26]. As ‘Industry 4.0 is only possible with the digital twin’ [27], identifying and resolving ambiguities in and closing the gap between standards and thereby fostering interoperability is of relevance for the success of industry 4.0.

The remainder of the paper is organized as follows. The classification scheme for IoT and DT standards is presented in Section 2. In Section 3, multiple standards from the domains IoT and DT are presented and analyzed based on the previously introduced classification scheme. This is followed by a summary including a comparison matrix discussing the findings and conclusions in Sections 4 and 5, respectively.

2. Classification Schema

Both IoT and DT center on resources. In the context of IoT, resources are internet-connected devices, whereby communication between consumer and device can happen either directly or via some kind of software system. In the context of DT, resources are defined in a broader sense, e.g., as assets, devices, and physical or virtual entities. Both concepts share the vision that communication between resources should happen mostly without human interaction, i.e., machine-to-machine (M2M).

Figure 1 depicts basic functions that are both required for realizing IoT and DT. As M2M communication requires a formal representation of the resource to interact with, the first step that needs to happen in such a scenario is always the creation of a resource description. A resource description is exactly that formal representation required and describes a resource with all its capabilities and interfaces that should be exposed to others. The creator of a resource description has the role of the provider. Once a resource description is created, it can be discovered by a consumer. A consumer can be a software system (i.e., M2M) or a human. Different modes of resource discovery can be supported, e.g., peer-to-peer (P2P) or directory-style (see Section 2.2). Once discovered, a consumer can parse/read the resource description and extract the information needed to access the described resource.

Although these are basic functions used in many other domains, these are the very functions enabling the IoT and DT to revolutionize many aspects of today’s IT landscape.

In the following, we present a classification schema for IoT and DT standards centered on the identified three main functions: resource description, resource discovery, and resource access.

2.1. Resource Description

Having a formal and machine-readable resource description is essential for IoT and DT, as it enables consumers to gain knowledge about available resources and defines how to interact with them. Resource descriptions are typically defined using a specific language or (meta) model, which can differ in many aspects, e.g., complexity, serialization formats, etc. In this paper, we discuss the following different classification criteria in the context of resource description.

2.1.1. Resource Term

What term is the central element of the language/(meta) model? Strictly speaking, this is not a real classification criterion but a helpful fact to find your way around the different terminologies used in different standards and platforms.

2.1.2. Model Type

Resource descriptions are expressed in the terms of a model (sometimes also referred to as a language). Depending on the expressiveness and generalizability, this model can be categorized as a different level following the Object Management Group (OMG) basic concept of multilevel metamodeling [28]. A detailed meta model hierarchy for IoT and DT is present in Figure 2. For each level, its content is described in the terms of the level above, whereas the top-level (M3) is typically self-describing. For example, a grounding in M3 could be RDF(S) (Resource Description Format Schema, optionally in combination with RDF Schema), M2 could be the Web Ontology Language (OWL), M1 could contain multiple (cross-)domain or application models like the Semantic Sensor Network (SSN) Ontology and M0 would contain the actual application data. M1 and M2 are often a hierarchy or network themselves built by re-using existing (and typically standardized) models combined with custom extension. In Figure 2, this is shown for M1 in an exemplary manner to reflect the hierarchical organization of models typically used in IoT and DT.

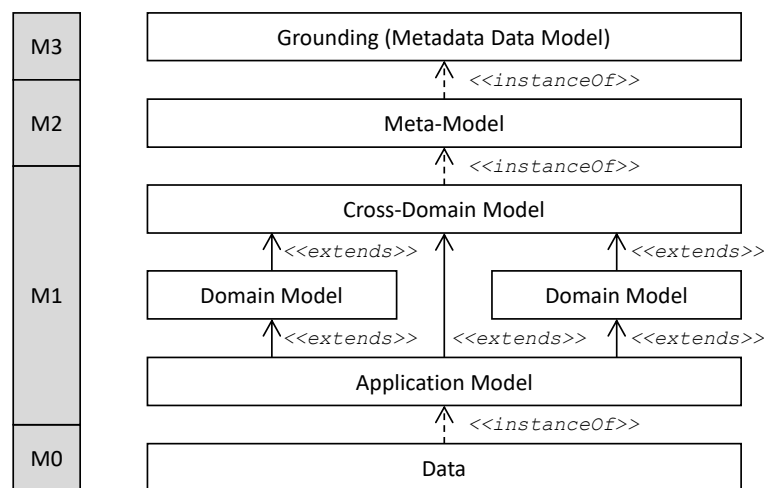


Figure 2. The metamodel hierarchy based on the Object Management Group basic idea of multilevel metamodeling [28] with an expanded M1 layer to reflect hierarchical organization of models typical for the internet of things and digital twins.

2.1.3. Resource Identification

Unique identification of any resource across system boundaries is essential in such highly distributed environments as IoT and DT (DTs may be passed along the supply chain along with (intermediate) products). Creating and keeping track of globally unique identifiers is not a trivial task and itself subject to standardization. Supporting different types of resource identifiers is also important when adding semantics through referencing external resources. Internationalized Resource

Identifiers (IRIs) are essential when working with ontologies as it is one of the basic building blocks of the Semantic Web. Other important external sources for semantics are international standards, e.g., eCl@ss [29], which often use International Registration Data Identifiers (IRDIs) to identify terms or concepts.

2.1.4. Type System

None of the considered standards provides a 'ready-to-use' application model but rather offer one or more models of a higher level (see Figure 2). This means that for a real-world application, a provider must define an application model based on the given higher-level models defining the concrete resource classes and their properties. This requires a type system defining the allowed/supported data types. Most standards re-use existing type systems, e.g., from XSD, JSON, or RDF, either directly or with slight restrictions and/or extensions.

2.1.5. Serialization Formats

To be exchanged between provider and consumer, a resource description needs to be serialized. This can be done in different formats like XML, JSON, JSON-LD, RDF or many others. Theoretically, most of these formats are interchangeable which also manifests in the fact that some standards support multiple serialization formats. However, the choice of serialization format can still influence other aspects, because e.g., a format might be coupled to a communication protocol or the type systems of the serialization formats might be difficult to match.

2.1.6. Resource Elements

The structure and expressiveness of a resource description language are one of the most important aspects of this classification. It defines what kind of information about a resource description can contain. The purpose of a resource description is to provide a consumer with all the necessary information to understand what this resource is and how to interact with it. It can be divided into two parts, one describing the capabilities of a resource and another one describing how to access it. The description of capabilities is very similar to the concept of interfaces in high-level programming languages and comes down to three types of elements: properties, functions, and events.

Properties refer to attributes of the resource and allow reading/writing or subscribing to them. Functions allow executing some logic on the resource supporting optional input and output parameters. In the context of IoT and DT, the term function is rarely used but instead the same concept is referred to as 'service'. Events are notifications about the occurrence of a state (change) or an action, e.g., 'battery level is critical' or 'button X has been pressed'.

2.1.7. Kinds of Data

The great variety of different use cases and application domains of IoT and DT require coping with different kind of data. The Big Data Value Association (BDVA) identified six different types of data, including time series, geo-spatial, temporal, and multimedia data, as well as the fact that different types of data require different query capabilities [30]. Adding explicit support for a special kind of data is something that is beyond the absolute core of a standard, i.e., it can be seen as adding a cross-domain main model according to Figure 2 below the meta-model (which is part of the absolute core). As most IoT and DT standards considered in this paper are either rather new or still being defined, they are still focusing on the core of the standards and often did not yet add explicit support for different kinds of data. In this paper, we, therefore, consider only the support for geo-spatial, historical, and temporal data as these are the most important aspect that are already covered at least by some platforms.

Geo-spatial data is typically needed when there is any interaction with the physical world, e.g., a resource with a physical location, typically somewhere on earth but in some scenarios, this could also be somewhere in space. Historical and temporal data is quite similar, as both require data to

be considered in the context of time. Historical data represent the history of a resource over time with a focus on the timely order of different states. Temporal data is considered simply data in the context of time. It can comprise data about the past but also the future of different resources at once, e.g., time-series data or results of prediction algorithms.

If standards support these different kinds of data is often determined by the paradigm followed. Probably most common is a device-centered paradigm where the virtual resource representing a physical device is considered a proxy reflecting the current state of the physical device as-is. Therefore, historical and temporal data is typically not available following this paradigm. Less common is an observation-based paradigm where observed changes in the environment are not only represented as changes in the state but also documented as an observation that was made by a sensor.

2.1.8. Resource Interlinking

As IoT and DT are highly dynamic environments of heterogeneous devices/resources, interlinking of resources essential to represent their (network) structure. The considered standards have different approaches and levels of support for resource interlinking which will be analyzed in detail for each standard in Section 3.

2.1.9. Semantic Annotation

In general, semantics is defined as ‘the meaning or relationship of meanings of a sign or set of signs’ [31]. In this context, signs are elements such as classes, properties, relations or instances of a knowledge base, expressed by their identifiers, e.g., <http://qudt.org/vocab/unit/#DegreeCelsius>. In technical systems, multiple levels of interoperability are to be considered [32]. Technical interoperability enables exchanging data on a physical level, syntactic interoperability that a system can parse the incoming data and semantic interoperability refers to ‘the ability of computer systems to exchange data with unambiguous, shared meaning’ [33]. As application models are not part of the standards, it cannot be expected that they are known to a consumer. Semantic annotation allows including references in the used (application) model(s), enabling consumers to discover the meaning of classes and properties on the fly.

2.2. Resource Discovery

For a consumer to know about existing resources, a discovery mechanism is needed. Discovering resources connected to a network is a common task, e.g., billions of resources are discovered via search engines every day.

Multiple well-established paradigms addressing resource discovery already exist, e.g., directories-style systems, search engines, crawlers, networking protocols (such as UPnP), etc. They can be divided into two groups; the ones defining how information about the existing resource is gathered and the ones defining how this gathered information could be accessed by users or other software systems. Information gathering can happen via e.g., (manual) registration of resources, networking protocols or crawlers. All of the considered standards default to manual or self-registration of a resource with some kind of resource directory. Some of them consider using special network protocols for resource discovery in the future but currently do not provide any solutions in this aspect. Therefore, we focus on how to access gathered information about existing resources, especially with the focus on supported network protocols and query capabilities.

2.2.1. Communication Protocols

Which communication protocols are supported for communication with the resource directory?

2.2.2. Query Capabilities

When trying to access a resource, a consumer typically does not want to access just any resource but a resource matching some defined criteria, e.g., measuring a certain property or offering a certain service. Although not always supported, it is a good idea for a resource directory to allow searching for resource matching certain criteria. In case searching is not supported, resource directories typically return the complete list of known resources, which then has to be filtered client-side. Therefore, our first classification criterion is whether queries generally are supported by the system.

In this paper, we focus only on the, from our perspective, most important aspects of query languages which are explained in the following. As query languages are often rather complex and differ in many small details, this set of classification criteria could be extended in the future. Because already many query languages exist, the most important criterion for us is which pre-existing query languages are supported. Typically, one pre-existing query language is chosen and used in a standard but some standards also support more than one or create a new one. Another criterion is if the query language(s) supports geo-spatial and temporal/historical queries. Although not directly affecting the query capabilities, we consider the possibility to add information about desired result formatting in a query a relevant criterion as it can massively improve usability and reduce network traffic when done right.

2.3. Resource Access

The actual goal in IoT and DT is accessing resources and communicating with them. Resource description and discovery are only intermediate steps required to achieve this goal. Of course, accessing and communicating with resources without underlying (IoT and DT) standards is also possible. However, this inevitably leads to writing lots of 'glue code', i.e., manually written code to communicate with different types of devices as they all use their own data formats and protocols. Especially in such a heterogeneous and dynamic environment like IoT where new devices are developed, connected to the internet and dynamically discovered by applications, writing glue code no longer is a viable solution.

2.3.1. Paradigm

We identified two different paradigms regarding resource access: API definition and API description. The most common paradigm to ensure unified resource access is to define some kind of resource API and require every resource to implement it. Although this is the easiest and most common approach, it does not take into account resources with existing APIs that cannot be updated to expose a new API. To properly address these issues and allow integration of legacy devices, the other paradigm is to not define a new resource API but rather provide a meta-model to describe existing APIs. On the downside, a lot of complexity is shifted to the client, i.e., implementing a client library becomes much harder as they now have to not only execute predefined commands via a single predefined protocol, but also understand the API description and potentially speak multiple protocols to communicate with a resource.

2.3.2. Communication Protocols

Resource access can happen via multiple different protocols, such as HTTP, CoAP, MQTT or OPC UA. Besides that, standards can be defined in a protocol-agnostic way and allow for mappings to other protocols to be added dynamically. This is especially important for standards using the API description paradigm.

3. Standards for Digital Twins and the Internet of Things

In this section, we introduce different standards related to TD and IoT and evaluate them according to the classification criteria defined in the previous section. The selection of standards was done based on discussions in different standardization bodies [34–36], recent scientific publications, as well as the

authors’ knowledge and experience in the areas of DT and IoT and their subjective relevance of the standards in practice. A previous work [5] focused on the extension of DTs with cognitive capabilities provides some additional insights to existing IoT and DT implementations.

3.1. Asset Administration Shell

The Asset Administration Shell (AAS) is a concept developed by the Plattform Industrie 4.0, a network of companies, associations, trade unions, science and politics in Germany [37]. It is part of the Reference Architectural Model Industry 4.0 (RAMI4.0) [38] and driven by the idea to manifest the concept of DT in factories and industrial production plants. The standard is divided into three parts. Part 1 [39] introduces the AAS meta-model and specifies how to serialize the content of an AAS, Part 2 defines how to interact with an AAS instance at runtime and Part 3 covers the infrastructure aspects and how to discover and interconnect multiple AASs [40]. This matches exactly our identified basic set of common functions: resource description, resource discovery, and resource access. At the time of writing, only Part 1 is officially released. Part 2 is available to the authors in a draft version.

An AAS is, just like a DT, a digital representation of a resource. Resources are referred to by the term ‘asset’, hence the name AAS. Figure 3 shows a class diagram of the AAS meta model. The main element is the AssetAdministrationShell, representing an Asset and composed of multiple Submodels. A Submodel is a container for SubmodelElements that can be properties, operations, collection, files, etc. The intention behind grouping sets of elements together into submodels and not adding those directly to an AAS is to allow standardization of submodels. By supporting multiple types of resource identifiers, IRI, IRDI and custom types, the standard provides much flexibility when referring to external resources e.g., for semantic annotations. The used type system is adapted from XSD complemented by the RDF data type langString. AAS descriptions can be serialized using XML, JSON, RDF, OPC UA data model and AutomationML. There is no explicit support for neither geo-spatial, temporal nor historical data although the standard is generic enough so this could be implemented on a higher level.

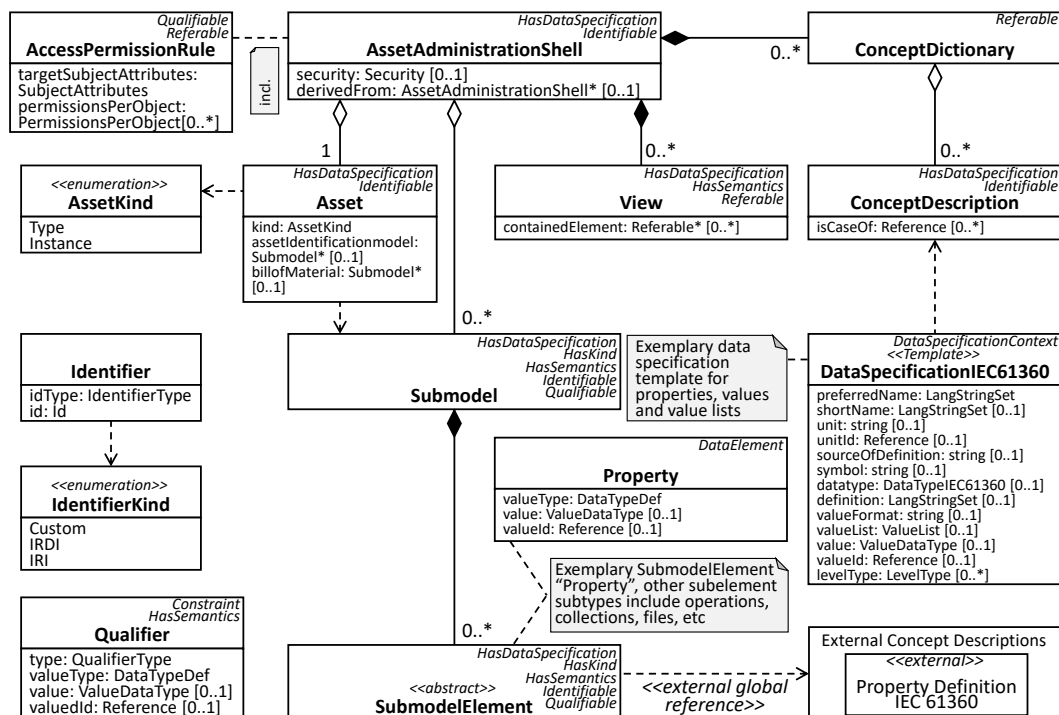


Figure 3. Class diagram depicting the metamodel of the AssetAdministrationShell concept developed by Plattform Industrie 4.0 (based on Figure 11 from [39]).

Resource discovery for this standard cannot be evaluated, as Part 3 is not yet released. However, Part 2 and [40] indicate that there will be an AAS registry to register AAS instances with some level of liability and a minimum set of metadata, e.g., endpoints. As an extension, AAS directories or even AAS catalogs may be built with further meta data elements that can be queried to discover resources.

For resource access, the standard will define a set of technology-specific APIs that each AAS instance has to implement. These APIs are derived from conceptual, technology-neutral interface specifications, which are currently being defined in Part 2. For the design of Industrie 4.0 systems and applications, they will be aggregated into services [41] by system and software architects, optionally extended with other, possibly proprietary interfaces. Therefore, the following information may be subject to changes in the future. At the core of the interfaces are basic CRUD (create, read, update, delete) operations for submodels and submodel elements. The interfaces will be defined for multiple communication protocols: HTTP(S), OPC UA, and probably MQTT.

3.2. Digital Twin Definition Language

The Digital Twin Definition Language (DTDL) [42] is developed by Microsoft and used in different products of their Azure services. Although not developed by an SDO, the DTDL is mentioned here because being already used in many commercial services offered by Microsoft like IoT Hub, IoT Central, and Azure Digital Twins. As the name suggests, the DTDL only covers the aspect of resource description and does not address resource discovery and resource access. There are two versions: Version 1 and Version 2. If not otherwise mentioned, we refer to Version 2 in this paper.

In the DTDL, resources are called interfaces and can contain a set of telemetry, properties, commands, relationship, and components as shown in Figure 4. For identification of resources and their elements DTDL uses a special form of URIs called Digital Twin Modeling Identifier (DTMI) of the form `<scheme>:<path>;<version>`. Telemetry describes data emitted by a resource in the form of a data stream. It corresponds to a combination of events and property subscriptions in our general terms. Commands correspond to functions that can be invoked with optional input and output parameters. Components are a similar concept as the submodels in the AAS providing a way to structure functionality in re-usable blocks. DTDL uses a custom type schema called Digital Twin Schema Definition Language, which is compatible with popular serialization formats, including JSON, Avro, and Protobuf. It supports complex interlinking of resources through explicit modeling of relationships including min/max cardinality. DTDL also supports semantic annotation for the type and unit of properties and telemetries. Unfortunately, only predefined semantic annotations are supported, i.e., extensibility is not given. JSON and RDF are supported in serialization formats. DTDL does neither support geo-spatial, temporal, nor historical data. Although in Version 1 a schema for geo-spatial data is defined (based on GeoJSON), it is not mentioned in Version 2 and therefore remains unclear if it still applies.

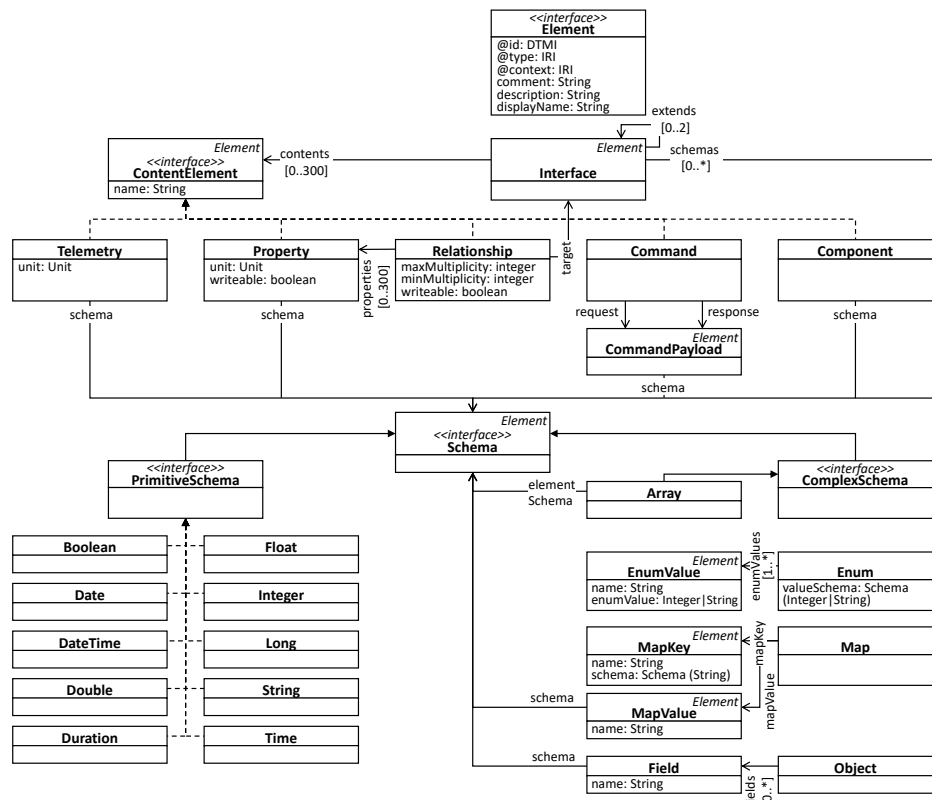


Figure 4. Class diagram depicting the metamodel of the digital twin definition language.

3.3. Next Generation Service Interfaces-Linked Data API

The Next Generation Service Interfaces-Linked Data (NGSI-LD) API [43] is an IoT standard defined by the ETSI Industry Specification Group (ISG) crosscutting Context Information Management (CIM). It is based on the Open Mobile Alliance (OMA) NGSI 9 and 10 interfaces [44] and FIWARE NGSIv2 [45]. In this paper, we refer to version 1.2.2 released 02/2020 [43]. Although never mentioning the term DT, the standard revolves around so-called ‘Entities represent[ing] physical or conceptual objects existing in the real world’ [46], which is more or less the definition of DT. In contrast to other standards, resources (entities) in NGSI-LD only comprise properties and relations to other resources. Services and events are not supported, although events can be partially realized through subscribing to property changes, which is supported.

As shown in Figure 5, NGSI-LD does not only provide a meta model defining general concepts such as Entity, Relationship and Property, but also a cross-domain ontology defining essential temporal and geo-spatial terms and concepts. Besides explicitly providing temporal and geo-spatial terms and concepts, the standard further supports historical data by allowing querying the state of an entity at any time in the past. Although, on a technical level, NGSI-LD could support resource identification via IRIs, they decided to allow only URIs ($URI \subset IRI$) for the sake of being compliant to the Linked Data principles [47]. Because of the focus on Semantic Web technologies and Linked Data principles, NGSI-LD provides good support for semantic annotations. Serialization happens in JSON-LD, which since version 1.1 is compatible to basic JSON as well as RDF.

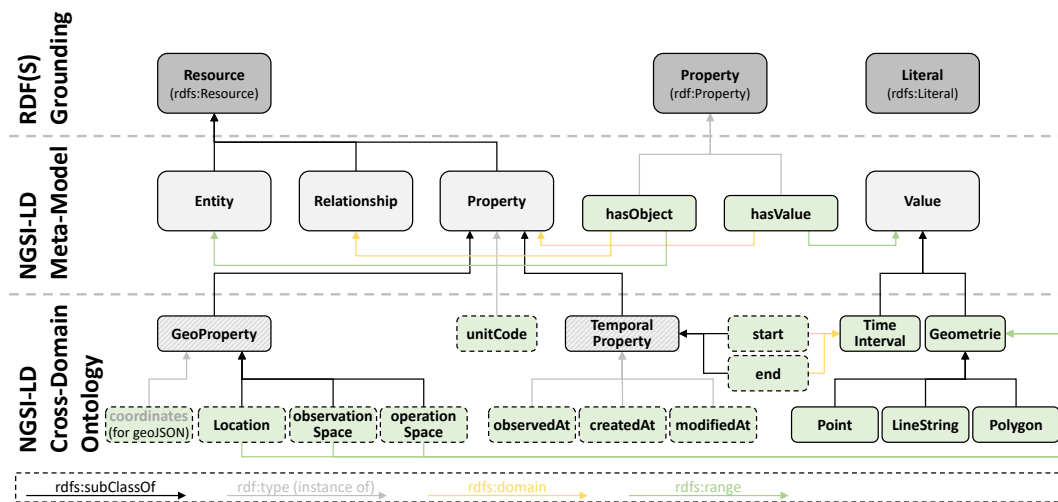


Figure 5. Class diagram depicting the metamodel hierarchy of the Next Generation Service Interfaces-Linked Data (NGSI-LD) API as defined by European Telecommunications Standards Institute (ETSI) (based on Figure 4.2.3-1 from [43]).

Resource discovery is realized by a component named ‘Context Registry’. NGSI-LD supports different types of architectures: centralized, distributed, and federated. In the centralized architecture, only one context registry exists whereas in the distributed and federated there can be multiple (hierarchically organized) registries. Any registry supports resource discovery via HTTP and custom query languages enabling geo-spatial, temporal and historical queries. Although there are already well-established query languages for RDF such as SPARQL or GeoSPARQL, NGSI-LD decided to define custom query languages from scratch. The reason for this is that NGSI-LD does not use the RDF data format where data is represented as triples in the form of <subject, predicate, object> but rather uses the concept of property graphs [48] where predicates of such triples can contain further key-value data.

Resource access in NGSI-LD is fundamentally different from the other standards as there is no direct communication between producer and consumer. Instead, the producer provides their information to a broker and consumers can then access these data or subscribe to be notified upon changes. These interactions are defined in a communication protocol-agnostic manner but currently only a binding to HTTP is provided.

3.4. Open Data Protocol

The Open Data Protocol (OData) [49] allows the definition of standardized but data model-agnostic RESTful APIs. It was initially developed by Microsoft in 2007 and Version 1.0, 2.0, and 3.0 were released under the Microsoft Open Specification Promise. In 2014, Version 4.0 was standardized by the Organization for the Advancement of Structured Information Standards (OASIS). This is the version we are referring to in this paper. Although not developed with IoT or DT explicitly in mind, we consider OData relevant for them as it is all about describing and accessing resources (even though originally resources were not considered physical).

The primary concept of OData is called ‘service’. Amongst others, a service comprises entity and entity set definitions as well as functions (must not have side effects, must have a result) and actions (may have side effects, may have a result). OData services are described by an Entity Data Model (EDM) expressed via the Common Schema Definition Language (CSDL), which in turn can be serialized via XML and JSON. The CSDL, therefore, corresponds to what we refer to as resource description in this article. The identification of resources happens primarily via URLs. Additionally, resources can also be identified via their entity type and ID, which, together with the URL conventions, implicitly defines the URL [50]. Resources can easily be interlinked and contain annotations, which can

be used to add (external) semantic. OData do not support defining events on resources, nor subscribing to property changes.

Similar to NGS-LD, there is no direct communication between producer and consumer in OData and all communication happens via a central server running the service. The standard defines a HTTP-based API for communication with the service that includes a powerful custom URL-based query language supporting geo-spatial filtering.

As all communication, resource access in OData happens via HTTP(S). URLs in OData are defined by the URL conventions in combination with the data model defined by the EDM. An example URL is shown in Figure 6. The service root part is given by the OData service and can be followed by any entity set name in the resource path. After that, any path defined by the EDM is allowed, e.g., accessing a single entity in an entity set by appending its ID in parenthesis as well as following relations or addressing properties by adding a followed by the name. The query options part of a URL can be used to specify the desired result more fine-grained.

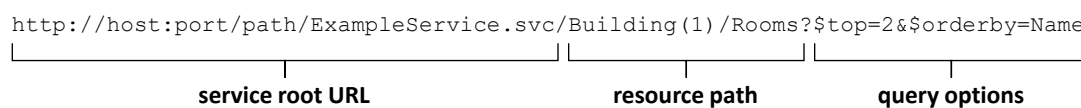


Figure 6. Components of URLs according to the Open Data Protocol specification.

3.5. SensorThings API

The SensorThings API (STA) standard [51] developed by the OGC SensorThings Standards Working Group (SWG) [52] provides a framework for interconnecting IoT devices, data, and applications over the Web. It is divided into multiple parts. Part 1 [53] was published in 2016 and covers the domain of sensing. Part 2 [54] addresses tasking, i.e., sending commands to devices, and was published in 2019. An updated version (v1.1) of Part 1 is finished and formal publication is underway.

Since 2001, the OGC is working on a suite of standards for managing sensors and actuators called OGC Sensor Web Enablement (SWE). It is designed to enable the connecting sensor and actuator to the Web and make them discoverable and query-able. As sensors and actuators are usually physical devices, geo-location is a key element of OGS standards. The older SWE standards, e.g., Sensor Observation Service (SOS) and Sensor Planning Service (SPS), use rather old technologies like XML and SOAP but more importantly, they are also missing some basic features like publish/subscribe (pub/sub) support or capabilities to update/delete data. STA addresses those issues and at the same time redesigns the APIs to fit modern web-based principles like REST-based services and JSON as payload format.

STA is strongly inspired by OData and re-uses some essential aspects, e.g., most parts of the URL conventions and query language, but is not fully compliant to OData. Besides leaving out some complexity of OData, STA also adds functionality where needed, e.g., pub/sub via MQTT and additional geo-spatial query capabilities. As OData itself is model-agnostic, STA provides a cross-domain model that can be further refined for any use case by providing an application model according to the metamodel hierarchy shown in Figure 2. The STA data model is shown in Figure 7. In STA, resources are called Things and are identified using URLs. The type system used is custom-built but re-using multiple existing type definitions, e.g., from previous SWE standards. Services are realized by modeling (virtual or physical) actuators with their capabilities and linking them to a thing. Events are only supported in the form of a subscription of changes of properties via MQTT. Resource interlinking and semantic annotation is hardly explicitly supported (only the class `ObservedProperty` has a property definition for semantic annotation) but can potentially be added by custom extension. Historical data is also only partially supported, as observations are stored over time, i.e., discovering past observations are possible but reading the value of properties at any time in the past is not possible.

Resource discovery works almost identically as in OData although STA provides enhanced geo-spatial query capabilities.

Resource access works also very similar to OData as both do not support direct communication between resource and consumer. In addition to OData, STA allows subscribing to the creation and update of resource collections, resources and properties. Having no direct communication between a consumer and a resource drastically improves usability in real-world applications, e.g., an actuator can subscribe to newly created tasks and immediately start processing them.

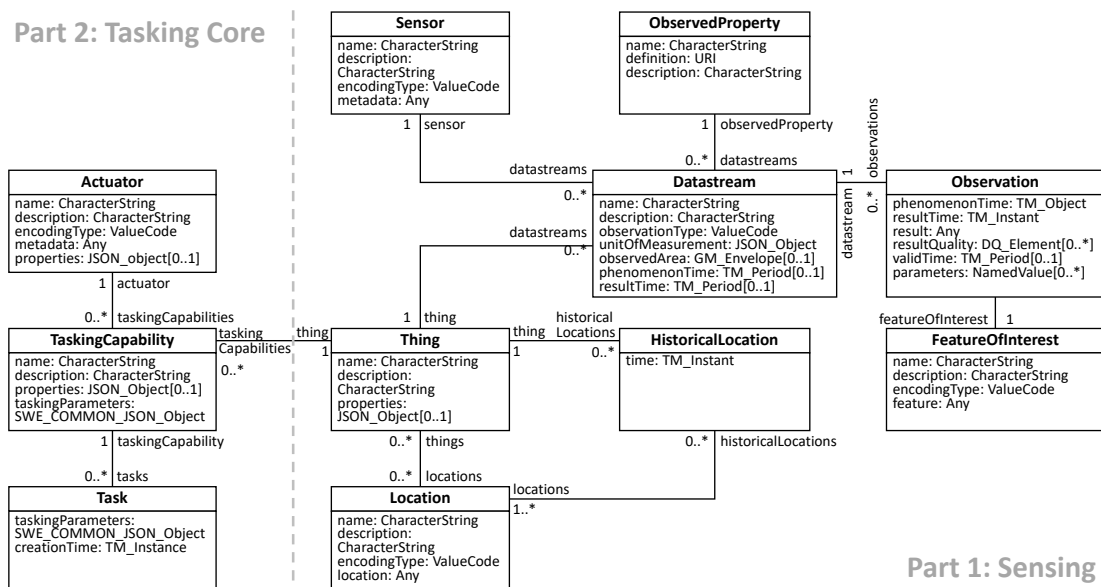


Figure 7. Class diagram depicting the data model of the SensorThings API Part 1 and 2 in version 1.0 (based on Figure 2 from [53] and Figure 1 from [54]).

3.6. Web of Things

The goal of the W3C Web of Things (WoT) Working Group is to counter the fragmentation of the IoT by providing so-called building blocks that should complete and enhance already existing standards. The most important of those building blocks is the WoT ThingDescription (WoT TD or only TD) [55], which became an official W3C Recommendation in April 2020. The TD provides a meta-model to describe existing resources in the IoT, e.g., a DT or a part of a DT, called Thing. As Figure 8 shows, a Thing comprises a set of properties (which can be read-only or read/write), actions, i.e., methods that can be invoked on the Thing, and events. The type system is based on JSON with additional support for adding constraints, e.g., regular expression for strings, based on JSON Schema. TDs use URI-based identifiers and can be serialized as either JSON-LD, JSON, or RDF. The standard does not contain any kind of (cross-)domain model and therefore neither supports geo-spatial, temporal nor historical data. One highlight of the TD is that by being grounded in Semantic Web technologies, it is easily possible to add semantic information to every aspect of a TD.

A serialized TD can be published and a consumer will be able to access the described resource. Resource discovery, i.e., how these TDs can be found, is still work in progress but is expected to be published in Q1 2021 [56]. Not yet standardized, there are also at least two existing implementations (<https://github.com/thingweb/thingweb-directory>) (<https://github.com/linksmart/thing-directory>) providing sophisticated search capabilities, e.g., via DNS-Based Service Discovery [57], CoRE Resource Directory [58], SPARQL queries or JSON-LD framing [59].

Unlike other standards, the TD does not define any API that has to be implemented by any resource. Instead, it defines a meta-model to describe existing APIs including e.g., protocol, payload format, security, in a machine-readable way. This is a unique and promising approach at closing the gap between existing (and future) standards and enabling interoperability.

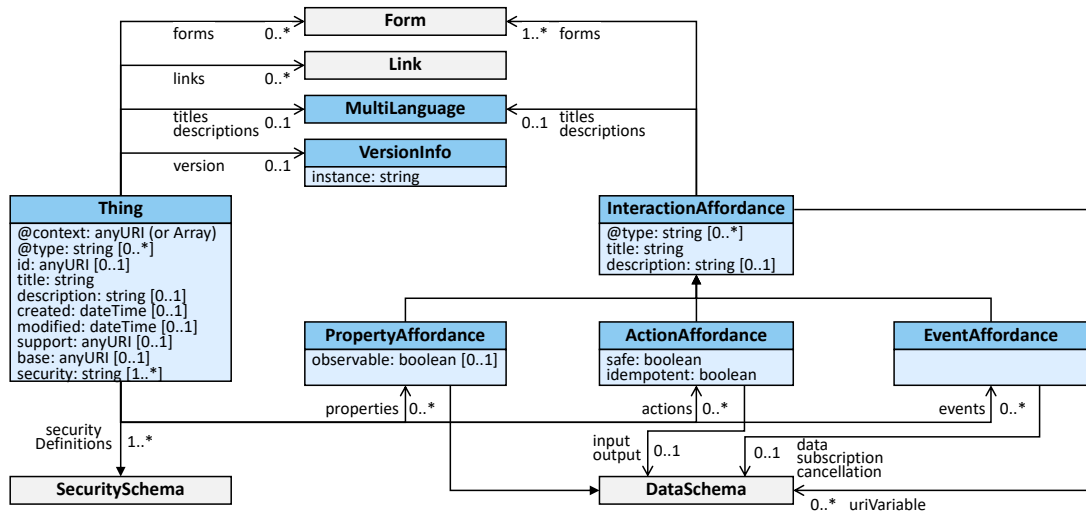


Figure 8. Class diagram depicting the metamodel of the Web of Things Thing Description (based on Figure 1 from [55]).

4. Summary

Table 1 provides an overview on the classification of the different IoT and DT standards analyzed. Although the standards were developed independently by different SDOs, they ended up with quite similar approaches/solutions to some aspects.

To enable resource description, providing a meta-model is the way to go. NGS-LD and STA also provide some cross-domain model, primarily defining terms needed for support of geo-spatial and temporal data. At first glance, the standards seem to use quite different types of resource identifiers. Considering that $IRI \supset URI \supset URL$, $URI \supset DTMI$, and $URL \cap DTMI = \emptyset$ it seems that URI might be a good compromise for a unified identifier type in the future. An even more flexible, but also more complex, approach is the one used in the AAS by defining meta-level constructs to describe any kind of identifier. All standards supporting resource interlinking (or at least having it on the agenda) reflects the fact that relations between resources are essential for IoT and DT. Besides, using different terms to describe the kind of elements that make up a resource, there seems to be more-or-less general consent that a resource should at least contain properties and services. Only half of the standards support resources defining custom events, but almost all support at least notification upon property changes. For serialization of resource descriptions, JSON is the common denominator between all standards. Since version 1.1, JSON-LD [60] supports interpreting basic JSON as JSON-LD when using HTTP and setting the header appropriately. Additionally, JSON-LD is compatible with RDF, the cornerstone of the Semantic Web. For this reason, almost all standards make use of JSON-LD to provide, on the one hand, easy-to-use JSON-based access and, on the other side, fine-grained access with full Semantic Web support. Again, the AAS provides a more generic approach supporting different kinds of serialization formats. This is partially because it also supports different protocols for resource access and some protocols require a certain serialization format, e.g., OPC UA.

Resource discovery is something on the agenda of most of the SDOs and will be properly addressed in the future. No standards currently supporting resource discovery allow direct communication between the consumer and the resource, i.e., they already have a central repository-like component by design. Furthermore, there is no consensus about which query language to use. Query languages are a complex subject itself. In fact, AAS and WoT are currently still evaluating different query languages that might be suited for them. Probably the main issue is the trade-off between the expressivity of a language and the complexity to implement it. Additionally, query languages are also often tied to specific data formats and/or protocols. SPARQL is a rather complex but powerful query language for RDF with SQL-like syntax. As most standards already support RDF data, SPARQL is one of the most promising candidates for a unified query language. Furthermore, there are already existing extensions for geo-spatial

queries in SPARQL, e.g., GeoSPARQL. Another promising approach are URL-based query languages like OData/STA and JSONPath. The query language of OData/STA has proven quite powerful in real-world applications but is much simpler to implement than SPARQL. JSONPath is adapting the concept of XPath from XML to JSON. It has been around for over ten years but has never been properly standardized by an SDO. Recently, efforts have been made initiated by the IRTF Thing-to-Thing Research Group (T2TRG) to pursue an official standardization and to push JSONPath as a query language in the IoT context [34,61].

Table 1. Comparison of the following internet of things and digital twin standards: AssetAdministrationShell (AAS), Digital Twin Definition Language (DTDL) v2.0, Next Generation Service Interfaces—Linked Data API (NGSI-LD API), Open Data Protocol (OData) v4.0, SensorThings API (STA) v1.0, and Web of Things (WoT).

	AAS	DTDL	NGSI-LD	OData	STA	WoT
Resource Description						
Resource Term	Asset	Interface	Entity	Entity	Thing	Thing
Model Type(s)	Meta	Meta	Meta	Meta	Cross-Domain	Meta
Resource Identification	IRI IRDI custom	DTMI	Cross-Domain URI	URL custom	URL custom	URI
Type System (based on)	XSD	custom	JSON GeoJSON JSON-LD	custom	JSON SWE-standards	JSON JSON Schema
Resource Interlinking	X	X	X	X	- ^a	X
Semantic Annotation	X	O ^b	X	-	O ^c	X
Resource Elements						
Properties	X	X	X	X	X	X
Services	X	X	-	O ^d	O	X
Events	X	X	O ^e	-	O ^e	X
Serialization Format	JSON RDF XML OPC UA AutomationML	JSON RDF Avro Protobuf	JSON RDF	JSON XML	JSON	JSON RDF
Supported Kind of Data						
geo-spatial	-	-	X	X	X	-
temporal	-	-	X	X	X	-
historical	-	-	X	-	O ^f	-
Resource Discovery						
Protocols	- ^a	-	HTTP	HTTP	HTTP	HTTP ^g CoAP ^g DNS-SD ^g O ^g
Querying supported?	- ^a	-	X	X	X	O ^g
Query Language						
Query Language based on	- ^a	-	custom	custom	OData	SPARQL ^{a,g}
geo-spatial queries	-	-	X	X	X	-
historical queries	-	-	X	-	O ^f	-
Resource Access						
API: Define vs. Describe	define	-	define	define	define	describe
Protocols	HTTP MQTT OPC UA	-	HTTP	HTTP	HTTP MQTT	HTTP MQTT CoAP
Protocols extendible?	-	-	X	-	-	X

^a extension under discussion; ^b only predefined definitions and only for telemetries, properties, and units; ^c only explicitly for observed properties and units, possible for everything else via custom properties; ^d only on service-level; ^e only property changes; ^f only for observations; ^g not part of standard, only in implementation(s); ^x y; Abbreviations: CoAP: Constrained Application Protocol; DNS-SD: Domain Name System - Service Discovery; HTTP: Hypertext Transfer Protocol; IRDI: International Registration Data Identifier; IRI: Internationalized Resource Identifier; JSON: JavaScript Object Notation; JSON-LD: JavaScript Object Notation - Linked Data; MQTT: Message Queuing Telemetry Transport; OPC UA: Open Platform Communications Unified Architecture; RDF: Resource Description Format; SPARQL: SPARQL and RDF Query Language; SWE: Sensor Web Enablement; URI: Uniform Resource Identifier; URL: Uniform Resource Locator; XML: Extensible Markup Language; XSD: XML Schema Definition.

Regarding resource access, the most prominent paradigm is to define a new API that each resource has to implement. The only standard adapting to another paradigm is WoT. Their idea is that there are already plenty of IoT devices and systems deployed and changing/updating their interfaces is not an option. Therefore, the WoT approach is to provide a way to describe existing resources and their APIs. Although WoT is the only standard following this approach, it seems like a valid claim considering there are already trillions of IoT devices deployed, often without the possibility to update them over-the-air. For resource access protocols, HTTP seems to be general consent. Typically, it is

accompanied by MQTT adding pub/sub functionality that HTTP is missing. Not limiting itself to a specific set of communication protocols seems like a good idea, as APIs should, in general, be protocol agnostic and also because the past shows that almost every communication protocol evolves over time and will eventually be replaced.

In general, it is to notice, that besides IoT and DT being broad domains with heterogeneous applications and partially different requirements, there exists some shared common core functionality of resource description, discovery and access. Different standards, developed by different SDOs, having different backgrounds like, e.g., environmental measurements, industrial production, appliances, or consumer electronics, came up with partially very similar standards, showing that there might be a chance for even better convergence. Furthermore, there is a need to abstract from the underlying technologies when mapping requirements on resource management and their DT counterparts systematically to capabilities of underlying IoT infrastructures [62]. DT Engineering will become a crucial methodology in future IoT environments.

5. Conclusions

The paper highlighted the relevance of IoT and DT and the importance of interoperability and standardization for industry 4.0. The description and managing of resources were identified as overlapping functionality between DT and IoT that is addressed by standards from both domains that need consolidation. This paper introduced a classification scheme based on the categories resource description, discovery, and access and presented relevant current standards from both domains. The standards were evaluated using the classification scheme and the results were compared. This analysis revealed commonalities and differences between the considered standards that can be used to stimulate the consolidation of standards in the future. Commonalities can essentially be found for basic aspects, e.g., that resources should be described by a set of properties, services, and events and that JSON-LD should be used as a serialization format for resource descriptions. There further seems to be an agreement that resource access through HTTP should be the default while additional protocols can be supported. Differences are typically related to more complex aspects, such as if geo-spatial, temporal, and historical data should be explicitly supported or which query language to use for resource discovery.

Some standards, especially the ones focused on DTs, are divided into multiple parts, and can be considered work in progress, as not all of those parts are published yet. The problem of competing and overlapping standards in these domains and the need for consolidation has already been acknowledged [35,63]. Cooperation between the involved SDOs is already happening but primarily on a working group level in a rather informal manner, e.g., by inviting representatives of other SDOs to present their work or persons that are active in multiple relevant working groups. This paper may be used as a starting point to steer this cooperation to a more formalized and technical level and thereby stimulate and advance the consolidation of standards in these domains.

In future work, the presented classification scheme can be extended and elaborated, e.g., with focus on semantic interoperability. Furthermore, additional standards may be added or existing ones updated as they evolve over time.

Author Contributions: M.J.: conceptualization, methodology, investigation, writing—original draft preparation, and visualization; T.U.: writing—review and supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly funded by the H2020 COGNITWIN project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870130.

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

Abbreviations

The following abbreviations are used in this manuscript:

AAS	Asset Administration Shell
AI	Artificial Intelligence
API	Application Programming Interface
BDVA	Big Data Value Association
CIM	Context Information Management
CoAP	Constrained Application Protocol
CoRE	Constrained RESTful Environments
CRUD	Create, Read, Update, Delete
CSDL	Common Schema Definition Language
CT	Cognitive Twin
DNS	Domain Name System
DT	Digital Twin
DTD	Digital Twin Definition Language
DTMI	Digital Twin Model Identifier
EDM	Entity Data Model
ETSI	European Telecommunications Standards Institute
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I4.0	Industry 4.0
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IIC	Industrial Internet Consortium
ISG	Industry Specification Group
IoT	Internet of Things
IoT-EPI	IoT-European Platforms Initiative
IRDI	International Registration Data Identifier
IRI	Internationalized Resource Identifier
IRTF	Internet Research Task Force
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
JSON	JavaScript Object Notation
LD	Linked Data
M2M	machine-to-machine
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NASA	National Aeronautics and Space Administration
NGSI	Next Generation Service Interfaces
OASIS	Organization for the Advancement of Structured Information Standards
OData	Open Data Protocol
OGC	Open Geospatial Consortium
OMA	Open Mobile Alliance
OMG	Object Management Group
OPC UA	Open Platform Communications Unified Architecture
OWL	Web Ontology Language
pub/sub	publish/subscribe
RAMI4.0	Reference Architectural Model Industry 4.0
RDF	Resource Description Format
RDFS	RDF Schema
REST	Representational State Transfer
SDO	Standards Developing Organization
SOAP	Simple Object Access protocol
SOS	Sensor Observation Service
SPARQL	SPARQL and RDF Query Language
SPS	Sensor Planning Service
SSN	Semantic Sensor Network (Ontology)

STA	SensorThings API
SWE	Sensor Web Enablement
SWG	Standards Working Group
T2TRG	Thing-to-Thing Research Group
TD	ThingDescription
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
XSD	XML Schema Definition
W3C	World Wide Web Consortium
WoT	Web of Things

References

1. Forbes. What Is Industry 4.0? Here's a Super Easy Explanation for Anyone. 2018. Available online: <https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/> (accessed on 9 August 2020).
2. Plattform Industrie 4.0. What Is Industrie 4.0? Available online: <https://www.plattform-i40.de/PI40/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html> (accessed on 9 August 2020).
3. Hermann, M.; Pentek, T.; Otto, B. Design principles for industrie 4.0 scenarios. In Proceedings of the 49th Hawaii international conference on system sciences (HICSS), Koloa, HI, USA, 5–8 January 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 3928–3937.
4. Malakuti, S.; van Schalkwyk, P.; Boss, B.; Sastry, C.R.; Runkana, V.; Lin, S.W.; Rix, S.; Green, G.; Baechle, K.; Nath, S.V. Digital Twins for Industrial Applications: Definition, Business Values, Design Aspects, Standards and Use Cases. In *Industrial Internet Consortium White Paper*; Industrial Internet Consortium: Milford, MA, USA, 2020.
5. Abburu, S.; Roman, D.; Berre, A.; Stojanovic, L.; Jacoby, M.; Stojanovic, N. COGNITWIN–Hybrid and Cognitive Digital Twins for the Process Industry. In Proceedings of the IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), 2020, Forthcoming.
6. Digital Twin Market Worth \$48.2 Billion by 2026. Available online: <https://www.marketsandmarkets.com/PressReleases/digital-twin.asp> (accessed on 9 August 2020).
7. Deloitte. Industry 4.0: Challenges and Solutions for the Digital Transformation and Use of Exponential Technologies. Available online: <https://www2.deloitte.com/content/dam/Deloitte/ch/Documents/manufacturing/ch-en-manufacturing-industry-4-0-24102014.pdf> (accessed on 9 August 2020).
8. Schleich, B.; Dittrich, M.A.; Clausmeyer, T.; Damgrave, R.; Erkoyuncu, J.A.; Haefner, B.; de Lange, J.; Plakhotnik, D.; Scheidel, W.; Wuest, T. Shifting value stream patterns along the product lifecycle with digital twins. *Procedia CIRP* **2019**, *86*, 3–11. [CrossRef]
9. Tao, F.; Zhang, M.; Nee, A.Y.C. *Digital Twin Driven Smart Manufacturing*; Academic Press: Cambridge, MA, USA, 2019.
10. Gartner, Inc. The Top 10 Technology Trends for 2012. Available online: <https://www.gartner.com/en/documents/1926316/the-top-10-technology-trends-for-2012> (accessed on 9 August 2020).
11. Gartner, Inc. The Top 10 Strategic Technology Trends for 2013. Available online: <https://www.gartner.com/en/documents/2335015/the-top-10-strategic-technology-trends-for-2013> (accessed on 9 August 2020).
12. Gartner, Inc. The Top 10 Strategic Technology Trends for 2014. Available online: <https://www.gartner.com/en/documents/2667526/the-top-10-strategic-technology-trends-for-2014> (accessed on 9 August 2020).
13. Gartner, Inc. The Top 10 Strategic Technology Trends for 2015. Available online: <https://www.gartner.com/en/documents/2964518/the-top-10-strategic-technology-trends-for-2015> (accessed on 9 August 2020).
14. Rivera, J.; van der Meulen, R. Forecast Alert: Internet of Things—Endpoints and Associated Services. 2017. Available online: <https://www.gartner.com/en/documents/3559634/forecast-alert-internet-of-things-endpoints-and-associat> (accessed on 9 August 2020).
15. Gartner, Inc. Top 10 Strategic Technology Trends for 2017. Available online: <https://www.gartner.com/en/documents/3471559/top-10-strategic-technology-trends-for-2017> (accessed on 9 August 2020).

16. Gartner, Inc. Gartner Top 10 Strategic Technology Trends for 2018. Available online: <https://www.gartner.com/en/documents/3867164/top-10-strategic-technology-trends-for-2018-digital-twin> (accessed on 9 August 2020).
17. Gartner, Inc. Gartner Top 10 Strategic Technology Trends for 2019. Available online: <https://www.gartner.com/en/documents/3904569/top-10-strategic-technology-trends-for-2019-digital-twin> (accessed on 9 August 2020).
18. Gartner, Inc. Gartner Top 10 Strategic Technology Trends for 2020. Available online: <https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020> (accessed on 9 August 2020).
19. Gartner, Inc. How Digital Twins Simplify the IoT. Available online: <https://www.gartner.com/smarterwithgartner/how-digital-twins-simplify-the-iot/> (accessed on 9 August 2020).
20. Xu, L.D. The contribution of systems science to Industry 4.0. *Syst. Res. Behav. Sci.* **2020**, *37*, 618–631. [CrossRef]
21. Burns, T.; Cosgrove, J.; Doyle, F. A Review of Interoperability Standards for Industry 4.0. *Procedia Manuf.* **2019**, *38*, 646–653. [CrossRef]
22. Lu, Y. Industry 4.0: A survey on technologies, applications and open research issues. *J. Ind. Inf. Integr.* **2017**, *6*, 1–10. [CrossRef]
23. Minerva, R.; Lee, G.M.; Crespi, N. Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models. *Proc. IEEE* **2020**. [CrossRef]
24. Global Digital Twins Market Insights 2020–2025 by Technology, Solution, Application and Industry Vertical. Available online: <https://www.globenewswire.com/news-release/2020/03/20/2003898/0/en/Global-Digital-Twins-Market-Insights-2020-2025-by-Technology-Solution-Application-and-Industry-Vertical.html> (accessed on 9 August 2020).
25. Transforma Insights. Global IoT Market Will Grow to 24.1 Billion Devices in 2030, Generating \$1.5 Trillion Annual Revenue. Available online: <https://transformainsights.com/news/iot-market-24-billion-usd15-trillion-revenue-2030> (accessed on 9 August 2020).
26. McKinsey Global Institute. *The Internet of Things: Mapping the Value Beyond the Hype*; McKinsey&Company: New York, NY, USA, 2015.
27. Digital Twin in the Industry 4.0: Interview with A Pioneer. 2018. Available online: <https://www.t-systems.com/de/en/about-t-systems/news/best-practice/03-2018-digital-twin/digital-twin-in-the-industry-4-0-interview-with-a-pioneer> (accessed on 9 August 2020).
28. *Meta Object Facility (MOF) Specification, Version 1.4*; Object Management Group, Ed.; Object Management Group: Milford, MA, USA, 2002.
29. eCl@ss: Classification and Product Description. Available online: <https://www.eclass.eu/> (accessed on 9 August 2020).
30. BDVA Strategic Research and Innovation Agenda (SRIA). 2017. Available online: https://bdva.eu/sites/default/files/BDVA_SRIA_v4_Ed1.1.pdf (accessed on 9 August 2020).
31. Merriam-Webster Dictionary: Semantics. Available online: <https://www.merriam-webster.com/dictionary/semantics> (accessed on 9 August 2020).
32. Wang, W.; Tolk, A.; Wang, W. The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S. In Proceedings of the Spring Simulation Multiconference, San Diego, CA, USA, 22–27 March 2009; p. 168.
33. Systems, Capabilities, Operations, Programs, and Enterprises (SCOPE) Model for Interoperability Assessment. Network-Centric Operations Industry Consortium, 2008. Available online: <http://www.jfsowa.com/ikl/scope08.pdf> (accessed on 9 August 2020).
34. IRTF Thing-to-Thing Research Group (T2TRG). Available online: <https://datatracker.ietf.org/rg/t2trg/charter/> (accessed on 9 August 2020).
35. Reach Out to Other Open Standards. Available online: <https://github.com/Azure/opendigitaltwins-dtdl/issues/10> (accessed on 9 August 2020).
36. Web of Things Working Group. Available online: <https://www.w3.org/WoT/WG/> (accessed on 9 August 2020).
37. Details of the Asset Administration Shell: From Idea to Implementation. Plattform Industrie 4.0, Ed. Available online: <https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/vws-in-detail-presentation.pdf> (accessed on 9 August 2020).

38. RAMI 4.0 - A Reference Framework for Digitalisation. Plattform Industrie 4.0, Ed. Available online: <https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.pdf> (accessed on 9 August 2020).
39. Details of the Asset Administration Shell: Part 1-The Exchanges of Information between Partners in the Value Chain of Industrie 4.0 (Version 2.0.1). Plattform Industrie 4.0, Ed. 2020. Available online: <https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1.pdf> (accessed on 9 August 2020).
40. Hoffmeister, M. Properties and the Asset Administration Shell as Information Backbone-How Can True Interoperability be Achieved along the Entire Industrie 4.0 Value Chain. Available online: https://www.eclass.eu/fileadmin/pdfs/1_16-55_Hoffmeister.pdf (accessed on 9 August 2020).
41. DIN SPEC 16593-1:2018-04 RM-SA-Reference Model for Industrie 4.0 Service Architectures-Part 1: Basic Concepts of an Interaction-Based Architecture; Usländer, T., Ed.; Beuth: Berlin, Germany, 2018. Available online: <https://www.beuth.de/en/technical-rule/din-spec-16593-1/287632675> (accessed on 9 August 2020).
42. Digital Twins Definition Language. Available online: <https://github.com/Azure/opendigitaltwins-dtdl/blob/master/DTDLD/v2/dtdlv2.md> (accessed on 9 August 2020).
43. ETSI GS CIM 009: NGSI-LD API. 2020. Available online: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.02.02_60/gs_CIM009v010202p.pdf (accessed on 9 August 2020).
44. Next Generation Service Interfaces Architecture. Open Mobile Alliance, Ed. 2012. Available online: http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-AD-NGSI-V1_0-20120529-A.pdf (accessed on 9 August 2020).
45. FIWARE-NGSI v2 Specification. Available online: https://www2.deloitte.com/content/dam/insights/us/articles/tech-trends-2020/DI_TechTrends2020.pdf (accessed on 9 August 2020).
46. ETSI GR CIM 008: Context Information Management (CIM); NGSI-LD Primer. 2020. Available online: https://www.etsi.org/deliver/etsi_gr/CIM/001_099/008/01.01.01_60/gr_CIM008v010101p.pdf (accessed on 9 August 2020).
47. Berners-Lee, T. Linked Data-Design Issues. 2006. Available online: <https://www.w3.org/DesignIssues/LinkedData.html> (accessed on 9 August 2020).
48. Defining a Property Graph. 2012. Available online: <https://github.com/tinkerpop/gremlin/wiki/Defining-a-Property-Graph> (accessed on 9 August 2020).
49. OData. Available online: <https://www.odata.org/> (accessed on 9 August 2020).
50. OData Version 4.01. Part 2: URL Conventions. 2016. Available online: <http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part2-url-conventions.pdf> (accessed on 9 August 2020).
51. OGC SensorThings API. Available online: <https://www.ogc.org/standards/sensorthings> (accessed on 9 August 2020).
52. SensorThings SWG. Available online: <https://www.ogc.org/projects/groups/sweiotswg> (accessed on 9 August 2020).
53. OGC SensorThings API Part 1: Sensing, Version 1.0; Liang, S., Huang, C.-Y., Khalafbeigi, T., Eds.; Open Geospatial Consortium: Wayland, MA, USA, 2016. Available online: <http://www.opengis.net/doc/is/sensorthings/1.0> (accessed on 9 August 2020).
54. OGC SensorThings API Part 2—Tasking Core, Version 1.0; Liang, S., Khalafbeigi, T., Eds.; Open Geospatial Consortium: Wayland, MA, USA, 2019. Available online: <http://www.opengis.net/doc/IS/sensorthings-part2-TaskingCore/1.0> (accessed on 9 August 2020).
55. Web of Things Thing Description. Available online: <https://www.w3.org/TR/wot-thing-description/> (accessed on 9 August 2020).
56. Web of Things Working Group Charter. Available online: <https://www.w3.org/2020/01/wot-wg-charter.html> (accessed on 9 August 2020).
57. Cheshire, S.; Krochmal, M. RFC 6763: DNS-based Service Discovery. 2013. Available online: <https://tools.ietf.org/html/rfc6763> (accessed on 9 August 2020).
58. Shelby, Z.; Koster, M.; Bormann, C.; van der Stok, P.; Amsuess, C. CoRE Resource Directory. 2017. Available online: <https://tools.ietf.org/html/draft-ietf-core-resource-directory-12> (accessed on 9 August 2020).
59. Longley, D.; Sporny, M.; Kellogg, G.; Lanthaler, M.; Lindström, N. W3C Recommendation: JSON-LD 1.1 Framing. 2020. Available online: <https://www.w3.org/TR/json-ld11-framing/> (accessed on 9 August 2020).

60. JSON-LD 1.1-A JSON-based Serialization for Linked Data. Available online: <https://www.w3.org/TR/json-ld11/> (accessed on 9 August 2020).
61. Gössner, S.; Normann, C. JSONPath–XPath for JSON. 2020. Available online: <https://tools.ietf.org/html/draft-goessner-dispatch-jsonpath-00> (accessed on 9 August 2020).
62. Usländer, T.; Batz, T. Agile service engineering in the industrial Internet of Things. *Future Internet* **2018**, *10*, 100. [[CrossRef](#)]
63. W3C and IIC Sign Liaison Agreement for the Industrial Internet of Things. Available online: <https://www.w3.org/blog/wotig/2015/08/11/w3c-and-iic-sign-liaison-agreement-for-the-industrial-internet-of-things/> (accessed on 9 August 2020).



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).