Extending the SensorThings API Data Model – Improving Interoperability and Use Case Flexibility in IoT

Arnold F. Arz von Straussenburg[1,*,†], Timon T. Aldenhoff[1,†] and Dennis M. Riehle[1,†]

[1]*University of Koblenz, Universitätsstraße 1, 56070 Koblenz, Germany*

Abstract

This study presents an enhancement to the OGC SensorThings API Data Model tailored for Internet of Things (IoT) environments, demonstrated with a Smart Farming application enhancement. The designed data model addresses critical challenges faced in real-world settings like industrial environments, adhering to the FAIR principles of Findability, Accessibility, and, in particular, Interoperability and Reusability. Beyond the practical use case of crop monitoring, we offer a conceptual framework for future projects across various domains. The resulting architecture demonstrates how modular components improve adaptability and extendibility through standardization and interoperability. This modular approach decouples modules such as device management from data storage, ensuring consistent data handling and supporting the integration and maintenance of diverse and evolving applications. The iterative development and evaluation process underlines the solution's effectiveness in managing IoT environments in practice. The findings highlight the potential for these extensible modules to be applied in other contexts, promoting a standardized yet flexible approach to IoT data management, supporting effective database design, and deriving best practices and design guidance for future projects. Additionally, our models approach IoT heterogeneity and interoperability, demonstrating clear advantages in modularity and standardized data handling, essential for managing complex, real-world IoT deployments in Smart Farming.

Keywords

Internet of Things, Smart Farming, Interoperability, Modular Data Model, OGC SensorThings API

1. Introduction

Since the 1990s, the Internet of Things (IoT) has been gaining prominence, with deployments increasing in interconnectivity, scale, and complexity [1, 2]. These architectures are applied in various fields, from urban environments to farming, where interoperability and data exchange capabilities are essential [3]. For instance, data platforms for urban communities or industrial environments integrate numerous components and extensive architectures [4], with standardized interfaces enabling seamless data sharing [5, 6].

Current IoT research spans multiple directions. A primary technological challenge involves standardizing IoT devices and communication mechanisms and integrating IoT data with organizational systems to enable real-world deployments in contexts like agricultural environments [7, 8]. Other challenges include organizational strategies for creating, capturing, and delivering value across IoT domains, individual lifestyle changes due to IoT, and societal efforts to examine policy regulations and security protocols to mitigate vulnerabilities [1]. Data platforms play a crucial role in IoT deployments, achieving interoperability through standardized data models that enable efficient data integration and sharing. Numerous projects from both research and practice implement IoT across various domains, such as Smart Cities and Smart Farming. Initiatives like the Smart City Marketplace [9] from the European Union (EU) exemplify the practical applications and guidance in the IoT area. Given the

ER2024: Companion Proceedings of the 43rd International Conference on Conceptual Modeling: ER Forum, Special Topics, Posters and Demos, October 28-31, 2024, Pittsburgh, Pennsylvania, USA

[*]Corresponding author.

[†]These authors contributed equally.

arz@uni-koblenz.de (A. F. Arz von Straussenburg); timonaldenhoff@uni-koblenz.de (T. T. Aldenhoff); riehle@uni-koblenz.de (D. M. Riehle)

https://uni-ko.de/riehle (D. M. Riehle)

0000-0001-9822-0586 (A. F. Arz von Straussenburg); 0009-0008-7929-4836 (T. T. Aldenhoff); 0000-0002-5071-2589 (D. M. Riehle)

complexity and scale of these projects, standards like the Open Geospatial Consortium (OGC) SensorThings API (STA) are being developed to impact their implementation positively [10, 7]. The Findability, Accessibility, Interoperability, and Reuse (FAIR) principles [11], which emphasize data reusability, have been established to enhance these efforts.

This research focuses on complex and critical smart applications like Smart Farming systems and irrigation control. Smart Farming leverages IoT technologies to optimize agricultural practices, enhancing productivity and sustainability. Smart Farming enables precise management of resources such as water and soil nutrients by integrating IoT devices for real-time monitoring and control. This approach not only increases efficiency but also addresses challenges like climate change and resource scarcity, making it a key area for innovation in agriculture. This research focuses on applying IoT solutions to Smart Farming, aiming to improve interoperability and adaptability in agricultural environments. It aims to integrate existing standardization efforts and the FAIR principles into an IoT architecture for real-world deployments. By leveraging existing standards, these deployments can achieve interoperability, adaptability, and maintainability while managing complex real-world applications effectively. Current standards in IoT primarily handle sensor data, often neglecting essential modules like device management, customer management, alerting, and failure detection [12]. Device management, for instance, is critical for overseeing numerous IoT devices [13], ensuring data storage, and managing the data sources, which is vital in cases of device failures or malfunctions [14, 15].

Many projects in Smart Cities or industrial contexts require these additional modules for effective data platform integration. However, most existing IoT platforms lack a standard data model, hindering the reuse and combination of data from different sensors [16, 15]. This results in nonstandard, situation-specific implementations that jeopardize interoperability, maintainability, and impede data sharing, failing to meet the FAIR principles. Introducing new technologies like time series databases requires proper structural integration. For practical IoT deployments, it is crucial to enable data platforms that ensure interoperability in Smart City deployments. To exemplify this, one can consider scientific data management, which aims to utilize IoT data for research and requires interoperability and device management. Thus, standardizing IoT devices and communication mechanisms presents a significant research opportunity in Information Systems (IS).

While existing models like the OGC STA have made strides toward addressing interoperability challenges, they often necessitate additional modules or extensions to function effectively in Smart City and industrial settings. This has led to a proliferation of diverse implementations and extensions around the STA, inadvertently compromising the FAIR principles. The resulting fragmentation hinders interoperability, maintainability, and the ability to combine data from different sensors, ultimately impeding the full potential of IoT deployments in crucial domains. In this work, we want to address this challenge of fragmentation and proliferation of extensions around the STA model.

The research objective (RO) of this study is to design a OGC STA-compatible data model extension architecture that incorporates the FAIR principles into IoT deployments, informed by a practical Smart Farming use case. Incorporating these principles, especially interoperability, resolves the outlined complications by providing a framework for optimizing data sharing and reusability as detailed by [11]. The implementation of the Smart Farming use case is the basis for the demonstration and evaluation. This study applies the OGC STA standard to an IoT application in Smart Farming, extending the architecture with modules like device management or alerting. It also creates abstractions for these modules and the various data sources. The resulting high-level architecture aims to guide future projects across different IoT domains by offering an interoperable approach to designing and integrating domain-specific modules based on the OGC STA data core. Leveraging experiences from a real-world Smart Farming deployment, this study provides insights that can be applied to other domains. This extends the higher-level, more abstract OGC STA data model. The study seeks to establish a standardized data model that simplifies interface mapping, enhancing interoperability and maintainability. The proposed architecture creates a modular, extensible framework focused on improving adaptability and extensibility through standardization, ensuring IoT deployments can efficiently manage and share data across various applications and environments.

The structure of this paper is as follows: Section 2 covers the relevant background on Smart City, Smart Farming, and the OGC STA. Section 3 presents an overview of related studies and projects. The research method is briefly explained in Section 4. Section 5 details the main results, focusing on the extension modules developed for the Smart Farming use case. Section 6 discusses the results, their generalizability, and limitations. Finally, Section 7 provides a conclusion and outlook for future research.

2. Background

2.1. OGC SensorThings API

The OGC STA is an established standard designed to seamlessly integrate various IoT devices, their associated data, and diverse applications within a unified web-based framework. This open and geospatially-enabled API facilitates the exchange of information between disparate components of the IoT ecosystem, thereby fostering interoperability and helping unlock the full potential of connected devices [10, 17, 18]. At its core, the STA encompasses two primary functionalities, Sensing and Tasking, and four more extensions. The Sensing part, hereafter called the *Sensing Module*, initially released as the foundational element of the STA [10], focuses on efficiently managing and retrieving observational data from heterogeneous IoT sensor systems.

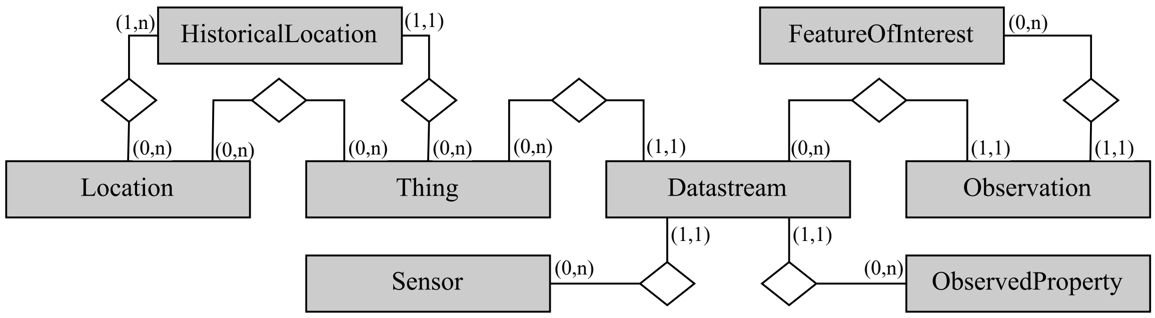

Figure 1: OGC SensorThings API Sensing Data Model (cf. [17]).

The *Sensing Data Model* comprises eight interconnected entities (Fig. 1). The central entity, *Thing*, represents a physical or virtual object under observation, such as an observed object. Each *Thing* is associated with one or more *Sensors* responsible for collecting *Observations*. Additionally, each *Thing* has a *Location*, which may change dynamically for moving objects, with *HistoricalLocations* tracking past positions. The data model allows multiple *Locations* to be linked to a *Thing*, accommodating diverse representations of the same physical location. *Datastreams* group *Observations* from a single *Sensor* measuring the same phenomenon, identified as an *ObservedProperty*. Each *Observation* is linked to a *Datastream* and a *FeatureOfInterest*, the specific aspect being observed. This interconnected model effectively captures the relationships between sensors, observations, and the objects and phenomena they represent, facilitating efficient data management and retrieval within IoT systems [10].

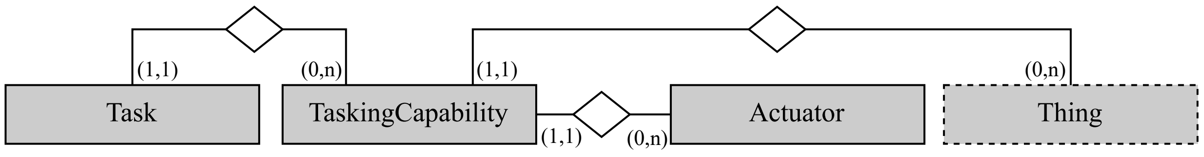

Figure 2: OGC SensorThings API Tasking Data Model (cf. [18]).

The second part of the OGC STA functionality, the Tasking part, in the following referred to as *Tasking Module*, has been introduced in the OGC STA standard after the Sensing part was established. The combination of Sensing and Tasking is thus considered the *Extended-STA* in some works [19, 20] and extends the data model of the *Sensing Module* with new entities (Fig. 2). The *Actuator* entity is central to this extension, representing a controllable device. The capabilities of an *Actuator*, describing its actions, are defined as *TaskingCapabilities.* The execution of a *TaskingCapability* is realized through a *Task*, linked to the capability and providing specific values for the *TaskingParameters.* The connection between the *Sensing* and *Tasking Module* is established through the shared *Thing* entity, allowing observation and control of the same physical object within the unified framework [18].

Beyond the Sensing and Tasking components, the STA offers four extensions to enhance its adaptability to different use cases. The *MultiDatastream Extension* [18] addresses the need to handle complex observations where results are represented as arrays, offering a structured approach for managing multidimensional data. The *Data Array Extension* [17] provides an alternative representation format for the *Observation* entities. It aggregates multiple observations into a data array. Furthermore, the communication capabilities are expanded with the *Sensing MQTT Extension* [17] and the *Tasking MQTT Extension* [18].

2.2. Smart Cities & Smart Farming

The concept of a Smart City encompasses a broad range of domains, including government, citizen services, business, and environment [2]. The IoT plays a central role in the environment domain in Smart Cities, enabling remote access to sensor data and control over physical elements, thus facilitating effective management of essential city resources (e.g., [21]). Notably, the environmental domain within Smart Cities intersects with the field of Smart Farming [3].

Smart Farming encompasses a variety of applications, including chemical control, crop monitoring, irrigation control, and more [3]. As with Smart Cities, different IoT solutions have been developed for monitoring crops, primarily focusing on collecting environmental data such as temperature, humidity, and luminosity. Furthermore, IoT solutions for irrigation control have been implemented in various agricultural settings. These solutions aim to optimize water usage in agriculture through diverse approaches, such as utilizing sensors to measure soil moisture and subsequently control irrigation sources, further optimizing the use of water [22, 23]. Another practical Smart Farming use case is described below.

2.3. Practical Use Case in Smart Farming

Our practical example, based on three years of experience with around 90 farms, mainly in Germany in the asparagus and strawberry sector, uses a similar system with some additional functions. Farmers purchase Narrowband IoT (Nb-IoT)-enabled devices and annual licenses for the primary strawberry and asparagus seasons in this use case. These devices are equipped with various sensors, e.g., temperature, humidity, and soil moisture sensors, deployed and utilized over the past three years. In addition, a planning phase for integrating actuators, such as motors for irrigation valves, into these devices is ongoing. The sensors from these devices are installed in the fields or greenhouses and transmit data via Nb-IoT technology to a highly available data platform [24] which serves as a central hub. It receives the continuous stream of raw sensor data transmitted via Nb-IoT, then performs several critical functions to transform this data into actionable insights. First, the platform processes the incoming data, organizing it into a structured format. Next, it cleans the data, removing any errors, inconsistencies, or outliers that might skew the analysis. Finally, the platform stores the cleaned data in a time series database, preserving the chronological order of measurements. The data is aggregated and optionally linked to weather data from external sources. Farmers can then access the extracted information via a mobile or web application. This allows them to monitor their crops remotely and receive alerts for irrigation needs, overheating, or frostbite. It is also possible to grant access to this information to their employees or other farmers.

3. Related Work

The increasing complexity and heterogeneity of IoT necessitate a focus on interoperability to ensure seamless communication and data exchange, as laid out in Section 1, and thus should be prioritized in developing new solutions [25]. Interoperability can be achieved by relying on accepted, widely used standards, incorped into reference architectures (e.g., [26, 24]). The OGC created several widely used geospatial standards and have gained acceptance and widespread use in various domains [16]. Among these standards, the STA has proven beneficial for managing and exchanging IoT sensor data [16]. It has garnered significant attention and adoption in academic research and practical applications. A quick search on the most extensive database for academic research papers, Scopus, revealed 40 papers mentioning the STA in their title, keywords, or abstracts since its inception in 2015, highlighting its traction in the research community. Furthermore, implementations of the STA have been integrated into various platforms from research and practice, including FRaunhofer Opensource SensorThings (FROST)-Server [27], BeAware [28], 52North [29], Go-SensorThings (GOST) [29], Mozilla [29], CGI Kinota Big Data [29], FIWARE [30], HERACLES [27], and INSPIRE [29], demonstrating its versatility and adaptability. The STA is also used to extend a platform for integration into the IoT, such as in the INSPIRE project [29].

Furthermore, the STA has been extended to integrate with other OGC standards, such as IndoorGML and CityGML [12]. In the context of smart cities, [19] and [12] proposed extensions to the STA data model, introducing the IndoorGML and CityGML entities like *Building*, *Room*, and *Opening* as types of *Things*. These integrations enable a more comprehensive representation of the built environment within the STA framework, facilitating the management and analysis of sensor data related to buildings and their components. Additionally, [31] explored the mapping of the STA onto the OpenIoT Middleware, further highlighting the potential for integration with other IoT standards and platforms. However, as [25] note, many standards, including potentially the STA, may face limitations due to being developed after real-world solutions, potentially resulting in insufficient breadth to address diverse situations.

Beyond these integrations, the STA offers official extensions to enhance its functionality and address specific use cases. These extensions include the MultiDatastream Extension, Data Array Extension, Sensing Message Queuing Telemetry Transport (MQTT) Extension, and Tasking MQTT Extension. Moreover, [20] proposed an extension to the STA ecosystem by introducing an automatic device registration process and extending its reach to the gateway and device layer. This extension aims to streamline device management and enhance the overall usability of the API. This addresses the growing complexity of device management in IoT, as highlighted by [25], who emphasize the need for abstracting data models and operations to simplify M2M communication and interoperability. Furthermore, including device management aligns with the requirements for IoT data platforms [7].

4. Method

To provide a solution to the research objective raised in Section 1, we derive a high-level data model for extending the OGC STA data model from a practical use case in the Smart Farming domain. The data model for this concrete Smart Farming IoT use case is created following the Design Science Research (DSR) methodology [32, 33, 34] and underwent three distinct iterations. For creating the higher-level data model extending the OGC STA, established mechanisms and concepts for Reference Architecture (RA) modeling and standardization, as outlined by [7], are incorporated.

The project trigger, motivated by practice, is the evolution of an existing NoSQL-based and normalized data model that highlighted the limitations of current practices in flexibility and scalability. This informed the first iteration of developing the demonstration use case, a transformation to a relational paradigm. The second iteration focuses on implementing the OGC STA standard [7, 17] to align with industry standards and enhance the structured handling of data. Moreover, this iteration laid the foundation for the integral interoperability framework and promoted standardization in the project.

The third and final iteration refined the architecture of the demonstratory use case by introducing modular components to improve adaptability and extensibility. These components were necessary based on real-world requirements in the practical deployment of the solution and subsequent evaluations. The resulting data model fulfills the goals of standardization and interoperability of IoT data faced in practice.

The design knowledge derived from the iterative development approach of the smart farming IoT application informs the developed conceptual model for designing more interoperable data models using the OGC STA in a standardized manner, presented in Section 6, which incorporates the design features identified in our demonstration [33] in Section 5. This framework, as presented in Section 6, as a meta-artifact, addresses the immediate technological challenges of standardization facing complex IoT deployments [1] and sets a foundation for further instantiations in various domains and use-cases.

5. Results

From the proposed practical Smart Farming use case (Section 2.3), we have been able to extract five major modules surrounding the *Extended STA*, as depicted in Fig. 3 as an overview: *Customer-*, *Device-*, *User Management* and *Alerting & Failure Detection*, as well as a *Data Source* module. Each module, except for the *Data Source* module, is decoupled from the *Extended STA*, ensuring flexibility and modularity. As illustrated in subsequent figures, all depicted interconnections are optional. Dotted borders around the depicted entities indicate a reference to another component. The central *Extended STA*, which includes both *Sensing* and *Tasking Modules*, is highlighted in gray in Fig. 3. The following figures highlight the entities that are part of the *Extended STA* in gray as well. The remaining entities, outlined with dots, reference the other developed components.

Except for Fig. 3, this paper presents entity–relationship (ER) models [35]. While a comprehensive data model was developed in practice, the focus in the following lies on illustrating the entities and their relationships rather than laying out the specific properties within each entity. Though these properties are relevant, they are beyond the scope of this work and do not directly contribute to our core arguments.

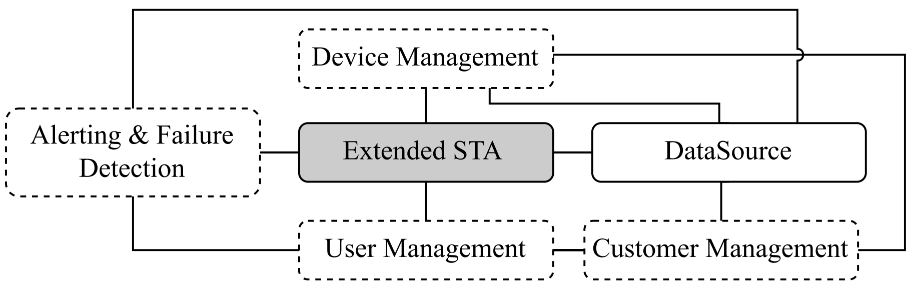

Figure 3: Overview of the identified extension for the SensorThings API.

While the *Data Source* component is essential for data inflow into the *Datastream*, it is not considered an extension like other modules. This distinction arises because any data, whether from a IoT *Device* or external APIs, like a *WeatherAPI*, inherently originates from a given source. The challenge of integrating diverse data sources has been highlighted in previous studies [16]. In practical scenarios, certain data sources are often linked to specific *Accounts*, a concept from the *Customer Management* module explored later in this section. The *Data Source* entity (Fig. 4) provides a mechanism to track the origins of data flowing into a given *Datastream*, which is part of the extended STA (c.f. Fig. 1).

Additionally, a *Trigger* entity can be associated with the *DataSource* to enable alerts, for example, when a *DataSource* becomes unavailable for a certain period, as described below in more detail. We argue that a *DataSource* is always necessary for a *Datastream* to function, as highlighted by the relationship depicted in Fig. 4. In contrast, the remaining relationships are all optional, indicating that a *DataSource* may or may not be associated with a *Device* or an external API if this kind of module is not used.

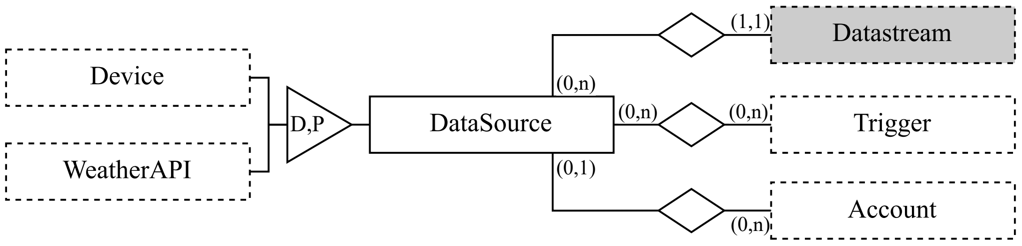

Figure 4: Data model of the Data Source module.

In the practical implementation, the device management challenge is a primary concern when integrating the STA. This challenge is documented in the literature [20] and is considered a fundamental requirement for IoT data platforms [7]. To address this, we developed a *Device Management* module (Fig. 5) tailored to our specific use case. The core entity in this module is the *Device*, which utilizes Nb-IoT technology. While a *SimCard* is typically associated with a *Device* for connectivity, a specific device might employ alternative technologies like Long Range (LoRa) that don't require a SIM card, but other *ConnectivityModules*.

Each *Device* is linked to a *DeviceType* to streamline device management. This entity acts as a schema, defining default metadata like manufacturer, connected sensors (*DeviceTypeSensor*), and standard values for sending and measurement intervals. This structure reduces redundancy by avoiding storing all metadata directly within each *Device* entity, especially when dealing with multiple similar devices.

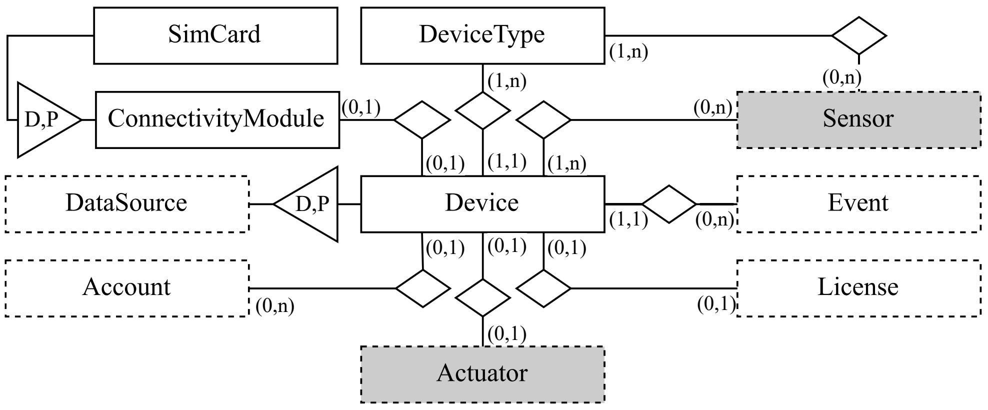

Figure 5: Data model of the Device Management module.

Within this framework, a *Device* represents a physical device equipped with a physical sensor. It is connected to *Sensors* from the *Extended STA*. For instance, an `ESP32` board with a `DS18B20` temperature sensor would be represented as a *Device* (ESP32) linked to a *Sensor* (DS18B20). Not all *Sensors* require a *Device*, as data from external APIs do not involve a physical device. Device events like firmware updates or setting changes are tracked using the *Event* entity. While these events could be managed within a time-series database in the *Sensing Module*, we opted for a separate entity to maintain a clear separation between *Device Management* and other aspects of the system.

To integrate *Customer Management* functionality, each *Device* is linked to *License* and *Account* entities. The *Account* entity represents the owner of a device. However, this relationship is marked as optional in the diagram to accommodate scenarios where a device might not have a designated owner. Additionally, a *Device* is associated with *Licenses*, which control whether data collection from a specific device is authorized and paid for. Further details on this mechanism are provided in the *Customer Management* module section below.

Furthermore, a *Device* is connected to one or more *Actuators* for tasking capabilities. This relationship enables the control of multiple actuators from a single device. An illustrative example is a device controlling multiple irrigation valves simultaneously.

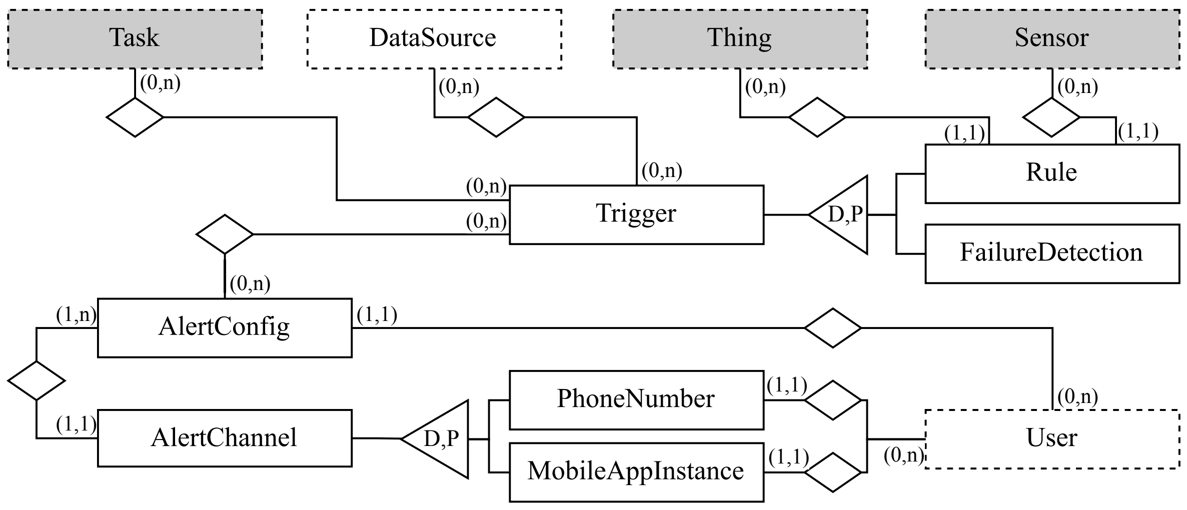

Figure 6: Data model of the Alerting & Failure Detection module.

The *Alerting & Failure Detection* module, illustrated in Fig. 6, is not a single module but instead comprises two distinct yet intertwined modules: the *Alerting* module and the *Failure Detection* module. Their combined representation in the diagram represents their structural similarity, as both serve as triggers for actions like *Tasks* or alerts through the *AlertConfig*. However, there is a fundamental difference between them.

The *Failure Detection* module focuses on monitoring the health and status of *DataSources* and, therefore, also potential *Devices*. Its core entity, *FailureDetection*, defines the conditions under which an alert should be triggered due to *DataSource* failure. For instance, an alert is triggered if a *WeatherAPI* or a *Device* with an expected 30-minute transmission interval fails to send data for an hour, as specified by the *FailureDetection* entity.

On the other hand, the *Rule* module provides a more generalized mechanism for triggering actions based on various conditions on sensor data. This allows alerts or *Tasks* to be triggered when a *Sensor* within a *Thing* from the *Sensing Module* meets the conditions of a predefined *Rule*. For example, in a greenhouse scenario, if a DS18B20 temperature sensor detects a temperature exceeding a threshold, an alert could be sent, or a task could be initiated to open windows for ventilation automatically. If a *Rule* is connected to a *Device* indirectly using the specified *Trigger* and *DataSource*, an on-device rule can be applied. One application could facilitate immediate data transmission when a physical device does a measurement or when a rule is triggered, even if the standard transmission interval is more prolonged. For example, a soil moisture sensor could trigger a transmission if the soil becomes too dry, regardless of the pre-set transmission schedule.

The *Trigger* entity acts as a bridge, connecting either *FailureDetection* or *Rule* entities to a specific *AlertConfig* or *Task* from the *Tasking Module*. This means that both device failures and rule violations can initiate actions. In our practical use case, an *AlertConfig* can be created by a *User* to configure an alert. This alert can then be triggered and utilize multiple *AlertChannels* to deliver the notification to the *User*. In this specific use case, a *User* is required for alerting, as the *AlertChannels* entity allows push notifications to be sent via a *MobileAppInstance* or calls and SMS messages via the *PhoneNumber* entity associated with a particular *User*. However, it is essential to note that other alerting methods not reliant on *Users*, such as sending API requests to external services, are also possible but fall outside the scope of our current implementation.

The alerting functionality motivates the creation of an integrated *User Management* in the considered use case for customized alerting and a basic permission system, allowing users access to specific *Things*. Fig. 7 illustrates the Data model for the *User Management* module. At the center is the *User* entity, connected to *AlertConfig*, *PhoneNumber*, and *MobileAppInstance*. This allows for personalized alert

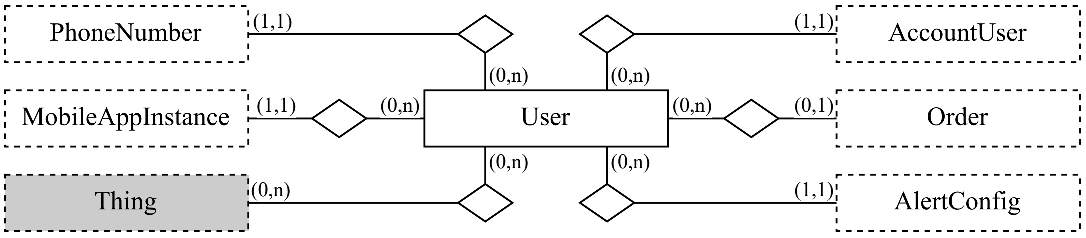

Figure 7: Data model of the User Management module.

delivery through preferred channels. The *Thing* entity demonstrates the connection to the *Extended STA*, enabling a rudimentary permission system where *Users* can be granted or denied access to specific *Things*. In the top right corner, there is a connection to the *Customer Management* module with the entities *Order* and *AccountUser*, facilitating potential integration with broader business processes and user account management.

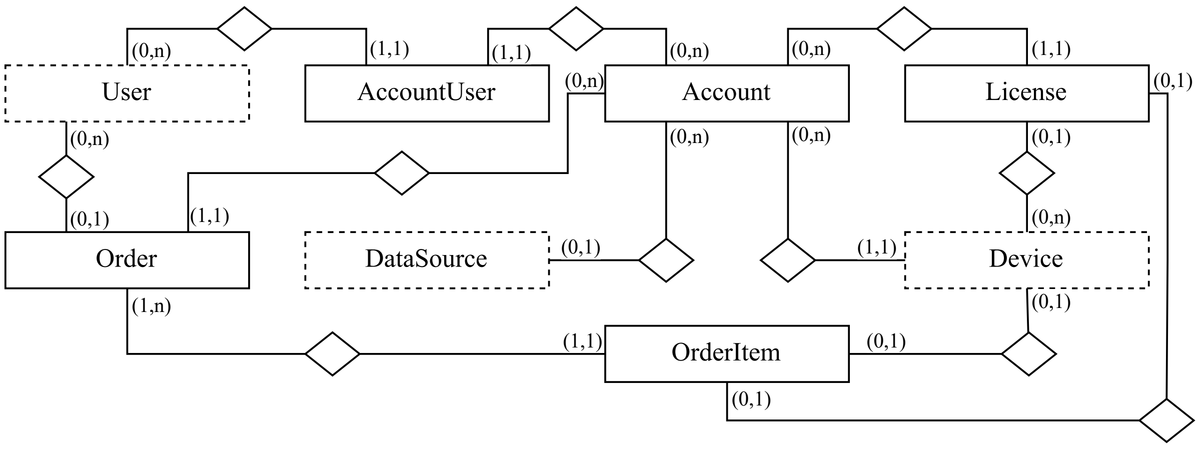

Figure 8: Data model of the Customer Management module.

The *Customer Management* module, shown in Fig. 8, is not directly connected to the *Extended STA* but interacts through the *Data Source* component. *Users*, such as employees of a farm, are associated with an *Account* via an *AccountUser* entity, which stores user permissions within the *Account*. In our Smart Farming use case, an *Account* represents a farm, and *Users* associated with that *Account* utilize the IoT data platform.

Users can create *Orders* with *OrderItems* for an *Account* to procure new *Devices* or *Licenses*. A *License* tracks whether an *Account* has paid for using a *Device* during a specific period. The *Account* is also linked to *DataSources* to provide custom *DataSources* created by *Accounts*. This structure ensures proper access control and billing management within the IoT data platform, aligning with the business rules of the practical Smart Farming context.

6. Discussion

By integrating the concepts for the OGC STA into the Smart Farming use case, both presented in Section 2, some of the problems formulated in Section 1 could be solved. Standardization, which can be achieved through the integration of established IoT architectures such as the OGC STA, has brought the artifact closer to fulfilling the formulated FAIR principles, but has not yet been able to solve all the challenges that have arisen in real-world deployments.

Several extensions were developed for the STA to address these identified challenges, presented in Section 5. Motivated by practical experience, these extensions solve problems that arise in the real-world deployment of the use case under consideration here, such as *Device Management*, shown in Fig. 5.

Some of these or similar challenges have already been identified in previous scientific studies or tackled in projects. For example, some projects mentioned in Section 2 are already solving challenges such as device management to enable real-world deployment. However, as this is very project-specific in the context of the various projects and quickly deviates from the standardized STA architecture, many of the advantages in interoperability, maintainability, and adaptability can be lost.

For this reason, all extensions implemented for our Smart Farming use case have been developed as independent modules that can interact with the STA in a standardized way. This is the result of the third design iteration of our artifact, and by incorporating the modular structure developed for this purpose, the OGC STA architecture can be progressively improved [32].

With the division of the STA architecture into the two main functionalities of sensing and tasking and thus the introduction of *Extended-STA*, OGC has already laid the foundation for the extensibility of the structure. In addition, with the technical extensions such as the *MQTT Extensions*, the *Data Array Extension*, and the *MultiData-stream Extension*, even more extensions have been considered that can be integrated into the STA. The extension modules presented here are based on these developments of STA, although they cannot yet be integrated into the existing extensibility structure.

While the technical extensions change the existing modules' behavior, the architecture's functional scope in the Extended-STA can be extended from sensing by implementing the tasking part. This becomes particularly clear through the extension of the data model, which represents the incorporation of tasking into the STA. However, the modules presented in Section 5 cannot simply be logically subordinated to the tasking module, as defined in the standard, of the *Extended-STA*.

The *Data Source* extension proposed in Fig. 4, integrating different and multiple data sources into the STA, does not represent *tasking* in its functional scope but is an independent type of extension. Data sources differ in form and structure from tasking modules and fulfill fundamentally different tasks. The same applies to the *management* modules for the *Device*, *User*, and *Customer Management* proposed in Section 5. These modules are similar in structure but differ from the *Tasking Module* described in the literature. In addition to the *Data Sources*, they represent a different type of extension to the STA basic structure than the *Tasking Moduls*. Finally, the situation is similar to *Alerting*, which is representative of several other conceivable monitoring modules. An example of another monitoring module is the *Failure Detection* of devices or data point transmissions.

Thus, we suggest extending the Extended-STA below so that other projects can benefit from the modules developed in this study based on our use cases. Formulating them as further extensions of the extenden-STA is helpful in the abovementioned manner. The extensions already implemented by other projects but not developed in a standardized way could thus benefit from interoperability, maintainability, and adaptability through a common architecture. New projects can be given guidance on how to tackle the identified real-world challenges. In addition to how interoperable modules are developed, the specific modules can provide helpful functionality for other projects.

For this reason, we propose the extensible structure from the *Existing STA*, *Data Source*, *Managing*, *Tasking*, and *Monitoring* modules as an additional extension, shown in Fig. 9. This model represents a higher-level conceptual model to the module overview presented in Section 5 and illustrates how the modular structure relates to the established core of OGC STA behaves. This is a deduction from the use case considered in this study, which can inform the design of further implementations.

The high-level concept shown in Fig. 3 represents the core of the architecture on the right-hand side - the *Sensing* module, which is extended below. As described above, outsourcing the data source is a sensible abstraction in many cases, as it connects multiple and different data sources. As at least one data source must always be defined, connecting *Data Sources* to the *Sensing* module is not optional.

The fact that several *Data Sources* can potentially be connected and that the list of *Manual Entries* (e.g., via a web interface or CSV ingestion), *Web Sources* (e.g., APIs), and *Devices* shown here is not complete is indicated by the disjunctive/partial specialization. It should also be emphasized that *Device Management* can be helpful when integrating (IoT-) devices.

In contrast to the mandatory *Data Sources*, it is also possible that no further *Extension Modules* are integrated into the architecture but that it only consists of the *Sensing* module. For this reason, it is

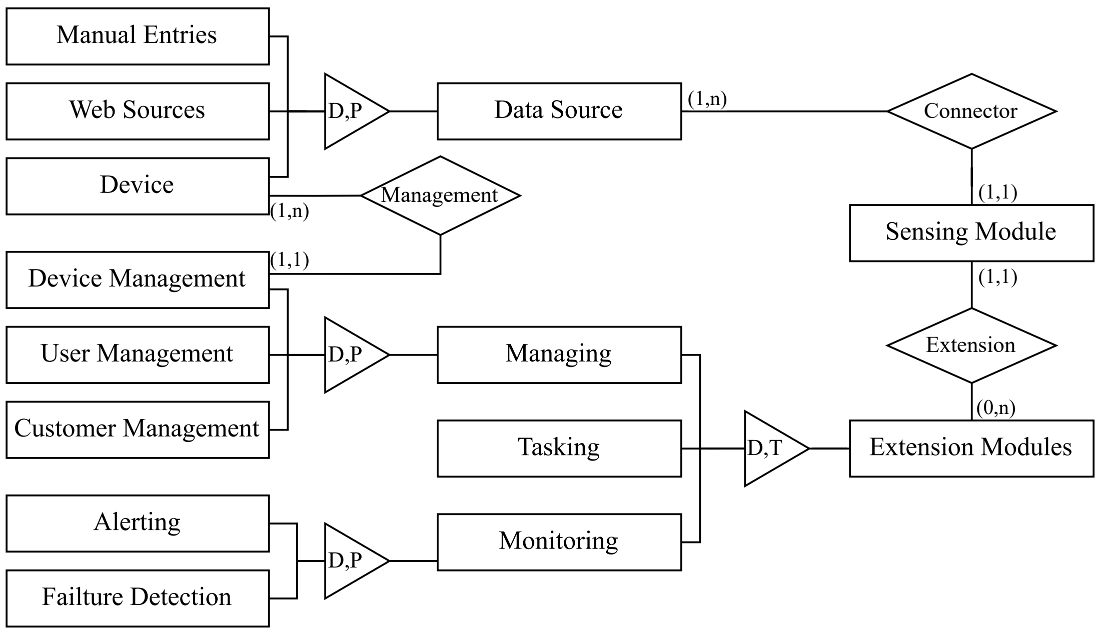

Figure 9: Higher level overview of the extensibility architecture.

optional for the extension relation connection to have an extension at all. As described above, *Extension Modules* can consist of three types: *Managing*, *Tasking*, and *Monitoring* extensions. As the specialization form disjoint/total indicates, this model does not provide for categories to be introduced beyond this to maintain a sufficiently high level of abstraction. The modules developed for this use case are examples in Fig. 3. It is also possible that further modules are informed by other use cases and projects, which is again indicated here by the disjunctive/partial specialization.

One of the best practices that could be identified is that the components should only have as few interdependencies as necessary, and should be optional if possible. One example is *Device Management*, which can introduce an optional interdependency for *Devices* if they are included as a *Data Source*. Furthermore, it has proven to be a productive approach to integrate more functionality by extending existing standards instead of implementing this in a non-standardized way. This ensures interoperability and reduces the implementation effort.

By successfully implementing the high-level architecture presented here for our Smart Farming use case, we have demonstrated its feasibility. We have shown how the extensible structure can contribute to getting closer to the implementation of FAIR principles. By establishing a standardized structure for the integration of extension modules, we improve the discoverability of data and functionalities within IoT ecosystems. The modular structure with clearly defined interfaces promotes accessibility and enables easier data retrieval and interaction across different platforms. In addition, the unified and extensible framework promotes interoperability by ensuring seamless communication between different modules and systems, effectively addressing a key challenge in heterogeneous IoT environments. Finally, the approach promotes modular development and reusability. Modules developed for specific use cases can be easily integrated into new projects, promoting efficient data exchange across different IoT implementations. Our extensions to the STA not only improve its functionality, but also align it with FAIR principles, promoting an open, accessible and reusable IoT data ecosystem.

Limitations of our study arise primarily from the fact that the data model results from the practical implementation of a single use case in the Smart Farming area. This is a common limitation, as has also been noted in previous studies [25]. As a result, the developed modules may not yet be complete and tailored to this specific use case. It is, therefore, conceivable that projects from other areas, especially from the field of Smart Cities or Smart Buildings/Smart Campuses, may find the modules presented here either less useful or incomplete.

Our higher-level conceptual model's abstraction level may also be unsuitable for other use cases. If only a single data source is connected, the integrated representation of the architecture of the OGC STA architecture as described in the standard, where the data source is integrated with the *Sensing* module, may be more suitable.

Furthermore, only the data model of STA, which underlies other aspects such as the REST API, was extended in this study. Although the extension of the REST API can be derived from this, it is not immediately apparent.

Existing IoT architectures have significantly impacted many projects in practice. This extension aims to enhance their utility by broadening their scope while maintaining flexibility. With the increasing proliferation of IoT and the development of more varied use cases, this extension might serve as a foundation for informing future design decisions. These proposed models and the modular architecture aim to increase interoperability, addressing the issue of IoT inhomogeneity by providing a structured approach. Creating a model emphasizing interoperability can mitigate the problems caused by diverse and incompatible IoT systems, ultimately fostering a more cohesive and efficient IoT ecosystem.

7. Conclusion

In this study, we created multiple extension modules and an extension to the OGC STA data model to control and track IoT devices. The modules' practical implementation in a Smart Farming application demonstrated their potential for real-world use, and the data model extension was built iteratively alongside it. The modular approach facilitates integration and maintenance while enhancing consistent data handling by separating components like device management from the data storage architecture. The enhanced data model architecture that results shows how using modular components can improve the application of FAIR principles. This technique can be applied to other areas that benefit from FAIR concepts, especially those related to interoperability and reusability, outside of Smart Farming. Our contributions include creating a modular, extensible framework that enhances data interoperability and device management in IoT deployments. This study demonstrates how the OGC STA can be extended to support the FAIR principles, particularly in improving interoperability and reusability in smart applications. The modular architecture enables better management of IoT devices and data sources, ensuring consistent and reliable data handling. The broader implications of this work extend beyond smart farming. The modular framework and the principles established here can be applied to various domains, including smart cities, healthcare, and environmental monitoring, where IoT deployments require robust data management and interoperability solutions. This generalizability underscores the versatility and potential impact of our proposed extensions.

While this study developed several key extension modules, future research should focus on developing additional modules and refining existing ones to address evolving needs. Implementing the OGC STA with our proposed extensions can guide future projects across diverse smart applications, like Smart City or Smart Farming. Existing projects using the *Extended OGC STA* could benefit from adopting our extension model to ensure interoperability and improve functionality. Additionally, there is potential for creating a platform-based approach for sharing interoperable modules, promoting a collaborative and standardized environment for IoT development. Addressing the lack of a standard data model in most IoT platforms, our proposed data integration step could facilitate the reuse and combination of data from different sensors, overcoming current limitations [16]. Our work advances IoT data model extensions by offering a scalable, interoperable, and flexible framework that aligns with the FAIR principles. This study establishes a foundation for future research and development, fostering innovation and promoting standardization in IoT deployments.

Acknowledgments

This research has been supported by the German Research Foundation (DFG) under Research Grant No. 432399058 and by the Federal Ministry of Education and Research (BMBF), Germany under Research Grant No. 16DTM218 as part of the NextGenerationEU program of the European Union.

References

[1] A. Baiyere, H. Topi, J. Wyatt, V. Venkatesh, B. Donnellan, Internet of Things (IoT) – A Research Agenda for Information Systems., Communications of the Association for IS 47 (2020) 21. doi:10.17705/1cais.04725.

[2] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, B. David, A literature survey on smart cities, Science China Information Sciences 58 (2015) 1–18. doi:10.1007/s11432-015-5397-4.

[3] E. Navarro, N. Costa, A. Pereira, A Systematic Review of IoT Solutions for Smart Farming, Sensors 20 (2020) 4231. doi:10.3390/s20154231.

[4] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, Y. Portugali, Smart cities of the future, The European Physical Journal Special Topics 214 (2012) 481–518. doi:10.1140/epjst/e2012-01703-3.

[5] G. Zhang, X. Liu, F. Zheng, Y. Sun, G. Liu, Geological disaster information sharing based on Internet of Things standardization, Environmental Earth Sciences 83 (2024) 148. doi:10.1007/s12665-023-11353-9.

[6] D. Correia, J. L. Marques, L. Teixeira, The State-of-the-Art of Smart Cities in the European Union, Smart Cities 5 (2022) 1776–1810. doi:10.3390/smartcities5040089.

[7] P. Fremantle, A reference architecture for the internet of things, WSO2 White paper (2015) 02–04.

[8] D. Marsh-Hunn, S. Trilles, A. González-Pérez, J. Torres-Sospedra, F. Ramos, A Comparative Study in the Standardization of IoT Devices Using Geospatial Web Standards, IEEE Sensors Journal 21 (2021) 5512–5528. doi:10.1109/jsen.2020.3031315.

[9] EU, Directorate-General for Energy, Investing in a sustainable and green urban future | Smart Cities Marketplace, https://smart-cities-marketplace.ec.europa.eu/, 2024.

[10] C.-Y. Huang, T. Khalafbeigi, OGC SensorThings API Part 1: Sensing, 2016.

[11] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, The FAIR Guiding Principles for scientific data management and stewardship, Scientific Data 3 (2016) 160018. doi:10.1038/sdata.2016.18.

[12] Y.-H. Chiang, C.-Y. Huang, M. Fuse, The Integration of OGC SensorThings API and OGC CityGML via Semantic Web Technology, in: S. Di Martino, Z. Fang, K.-J. Li (Eds.), Web and Wireless Geographical Information Systems, volume 12473, Springer International Publishing, Cham, 2020, pp. 55–67. doi:10.1007/978-3-030-60952-8_6.

[13] M. Blazevic, T. T. Aldenhoff, D. M. Riehle, Towards a Smarter Tomorrow: A Design Science Perspective on Building a Smart Campus IoT Data Platform, in: M. Mandviwalla, M. Söllner, T. Tuunanen (Eds.), Design Science Research for a Resilient Future, Springer Nature Switzerland, Cham, 2024, pp. 262–277. doi:10.1007/978-3-031-61175-9_18.

[14] H. van der Schaaf, J. Moßgraber, S. Grellet, M. Beaufils, K. Schleidt, T. Usländer, An Environmental Sensor Data Suite Using the OGC SensorThings API, IFIP Advances in Information and Communication Technology 554 Ifip (2020) 228–241. doi:10.1007/978-3-030-39815-6_22.

[15] M. Blazevic, D. M. Riehle, Comparing On-Premise IoT Platforms: Empowering University of Things Ecosystems with Effective Device Management:, in: Proceedings of the 9th International Conference on Internet of Things, Big Data and Security, SCITEPRESS - Science and Technology Publications, Angers, France, 2024, pp. 310–320. doi:10.5220/0012727000003705.

[16] P. Hertweck, T. Hellmund, J. Moßgraber, Opal—the toolbox for the integration and analysis of iot in a semantically annotated way, Sensors 21 (2021). doi:10.3390/s21124002.

[17] S. Liang, T. Khalafbeigi, H. Van Der Schaaf, B. Miles, K. Schleidt, S. Grellet, M. Beaufils, M. Alzona, OGC SensorThings API Part 1: Sensing Version 1.1, 2021.

[18] S. Liang, T. Khalafbeigi, OGC SensorThings API Part 2 – Tasking Core, Version 1.0., 2019. doi:10.25607/obp-454.

[19] C.-Y. Huang, Y.-H. Chiang, F. Tsai, An ontology integrating the open standards of city models and Internet of things for smart-city applications, IEEE Internet of Things Journal 9 (2022) 20444–20457.

[20] C.-Y. Huang, H.-H. Chen, An Automatic Embedded Device Registration Procedure Based on the OGC SensorThings API, Sensors 19 (2019) 495. doi:10.3390/s19030495.

[21] D. M. Riehle, A. F. Arz von Straussenburg, M. Blazevic, A. Wolters, Sensor-based Use Case Advancements in Smart Parking - Pioneering the Next Generation of Smart City Applications, in: Proceedings of the Australasian Conference on Information Systems, ACIS 2023, 2023.

[22] L. M. Fernández-Ahumada, J. Ramírez-Faz, M. Torres-Romero, R. López-Luque, Proposal for the design of monitoring and operating irrigation networks based on IoT, cloud computing and free hardware technologies, Sensors 19 (2019) 2318.

[23] I. Mohanraj, K. Ashokumar, J. Naren, Field monitoring and automation using IOT in agriculture domain, Procedia Computer Science 93 (2016) 931–939.

[24] T. T. Aldenhoff, A. F. Arz von Straussenburg, D. M. Riehle, Designing for high availability – A reference architecture for IoT data platforms, in: Proceedings of the 28th Pacific-Asia Conference on Information Systems (PACIS), Ho Chi Minh City, Vietnam, 2024.

[25] V. S. Kumar, S. Kulkarni, N. Mukkapati, A. Singhal, M. Tiwari, D. S. David, Investigation on Constraints and Recommended Context Aware Elicitation for IoT Runtime Workflow, International Journal of Intelligent Systems and Applications in Engineering 12 (2024) 96–105.

[26] M. Blazevic, D. M. Riehle, Enabling smart collaborboration spaces in organizations: Foundations and an integrated IoT architecture., in: Proceedings of the 28th Pacific-Asia Conference on Information Systems (PACIS), Ho Chi Minh City, Vietnam, 2024.

[27] P. Hertweck, T. Hellmund, H. van der Schaaf, J. Moßgraber, J.-W. Blume, Management of Sensor Data with Open Standards., in: Iscram, 2019.

[28] G. Antzoulatos, A. Karakostas, S. Vrochidis, I. Kompatsiaris, F. Lombardo, D. Norbiato, M. Ferri, A Crisis Classification System for flood risk assessment: The beAWARE project, in: 2nd International Conference Citizen Observatories for Natural Hazards and Water Manageme, Zenodo, Venice, 2018-11-27/2018-11-30. doi:10.5281/zenodo.3739200.

[29] A. Kotsev, K. Schleidt, S. Liang, H. van der Schaaf, T. Khalafbeigi, S. Grellet, M. Lutz, S. Jirka, M. Beaufils, Extending INSPIRE to the internet of things through SensorThings API, Geosciences 8 (2018). URL: https://www.mdpi.com/2076-3263/8/6/221. doi:10.3390/geosciences8060221.

[30] F. Cirillo, G. Solmaz, E. L. Berz, M. Bauer, B. Cheng, E. Kovacs, A Standard-Based Open Source IoT Platform: FIWARE, IEEE Internet of Things Magazine 2 (2019) 12–18. doi:10.1109/iotm.0001.1800022.

[31] H. van der Schaaf, R. Herzog, Mapping the OGC SensorThings API onto the OpenIoT Middleware, in: I. Podnar Žarko, K. Pripužić, M. Serrano (Eds.), Interoperability and Open-Source Solutions for the Internet of Things, volume 9001, Springer International Publishing, Cham, 2015, pp. 62–70. doi:10.1007/978-3-319-16546-2_6.

[32] A. Drechsler, A. Hevner, A four-cycle model of IS design science research: Capturing the dynamic nature of IS artifact design, in: Proceedings of the DESRIST 2016, Desrist 2016, St. John, Canada, 2016.

[33] A. R. Hevner, S. T. March, J. Park, S. Ram, Design Science in Information Systems Research, MIS Quarterly 28 (2004) 75–105. doi:10.2307/25148625.

[34] M. T. Mullarkey, A. R. Hevner, An elaborated action design research process model, European Journal of Information Systems 28 (2018) 6–20. doi:10.1080/0960085x.2018.1451811.

[35] P. P.-S. Chen, The Entity-Relationship Model — Toward a Unified View of Data, ACM Transactions on Database Systems 1 (1976) 9–36.