Hindawi
Mobile Information Systems
Volume 2021, Article ID 5535231, 18 pages
https://doi.org/10.1155/2021/5535231

*Research Article*

# An Extensible Data Enrichment Scheme for Providing Intelligent Services in Internet of Things Environments

**Yoosang Park, Jongsun Choi, and Jaeyoung Choi**

*School of Computer Science and Engineering, Soongsil University, 369 Sangdo-Ro, Dongjak-Gu, Seoul 06978, Republic of Korea*

Correspondence should be addressed to Jaeyoung Choi; choi@ssu.ac.kr

Received 25 February 2021; Accepted 4 May 2021; Published 30 May 2021

Academic Editor: Lidia Ogiela



Recent technologies in the Internet of Things (IoT) environment aim to provide intelligent services to users. Intelligent services can be managed and executed by systems that handle context information sets. Handling intelligent services leads to three major considerations: objects in the real world that should be described as metadata, a data enrichment procedure from sensing values for representing states, and controlling functionalities to manage services. In this study, an extensible data-enrichment scheme is proposed. The proposed scheme provides a way to describe profiles, data abstraction procedures, and functionalities that support the building of context information sets derived from raw datasets in the manner of a semantic web stack. Finally, data enrichment will help any system that uses context information by providing improved, understandable, and readable datasets to the service developers or the systems themselves.

## 1. Introduction

With the advent of Internet technology, the term "Internet of Things (IoT) environment" broadly applies to diverse industries and domains, such as agriculture, robotics, and autonomous vehicles [1–3]. However, there are three major concerns to be considered when using IoT techniques to provide intelligent services.

First, objects in the real world should be described in a unified way to be accessed by the users and relevant systems. This can lead to resolving heterogeneous issues; for example, developers still exert extra effort into developing source code interfaces as manufacturers provide different levels of sample codes for various purposes [4–6]. Besides, there are common factors to be considered when working with these devices, such as data types, data abstraction methods, and data representations. These factors are selected to enable life cycles that are visible and manageable based on service requirements, time spent by developers and users, and expense costs.

Second, the status values should be properly matched. The term status value implies how users, service developers, and systems can understand the sensing values from the devices, which are represented in different formats. For example, commercial sensor devices may provide readable values or string literals that users and systems can understand. However, limited approaches have been conducted regarding this as diverse sensor devices or relevant devices in the middle of the implementation process do not fully provide proper code blocks to manipulate and process functionalities or sensing values [7–10].

Third, controlling functionalities should be considered to provide built-in or ready-to-execute services by a sequence of service names, input parameters, output responses, and their data types. Consider context-aware systems that provide services based on the surroundings of deployed sensors and robotic devices as a practical implementation case. When robots are about to proceed to activate the services, there should be descriptions of functionalities written in certain languages. This third issue has several perspectives regarding controlling functionalities that are directly connected to the output interfaces for developers, users, and relevant systems. Therefore, these conflicting requirements should be considered among the people or developers who are interested in the phases of development, service planning, and avoidance of

malfunctions [11–13]. One advantage of considering functionality controls in the robotics domain is that recent applications and submodules are run and managed by Robot Operating System (ROS). However, software developers who desire to use these functionalities may not have sufficient experience in manipulation because learning detailed knowledge of operating robots is necessary. Therefore, intercommunication modules should be provided to developers to meet these requirements.

In this study, we propose a new scheme called ThingsMetadata to address these issues as an XML implementation of the semantic web stack. ThingsMetadata enables both software developers and robotics engineers to describe objects in the real world as metadata, for example, profiles, sensor devices, and robotic devices. It provides basic information and processing routines for representing them into Resource Description Framework (RDF) triplet called a data enrichment procedure. Data enrichment proceeds to change the gathered values from the sensor devices to an equivalent level of status information. Finally, it allows the developers to define the types of deployed robot functionalities, and these functionalities can be utilized as metadata to provide the services. Applying the proposed scheme provides descriptions to build data enrichment procedures, including meta datasets for physical objects in the real-world scene, accessing devices to collect sensing values, processing the values to the user-defined state that service developers can use them to describe service scenario documents, and representing result datasets which contain the current state of sensing value in a RDF format.

## 2. Literature Review

The materials and methods section should contain sufficient detail so that all procedures can be repeated. It may be divided into headed subsections if several methods are described. When systems work with IoT environments to provide intelligent services, there are three concerns to be considered.

The first is the openness of resources, objects, and services. Openness provides better visibility for discovering, accessing, and deploying resources to the users and systems. Studies on openness have focused on building a set of knowledge bases for decades [14–18]. These studies mainly consider a concept called metadata as a key component of building a knowledge base for defining objects characteristics. Certainly, there can be different formats or structures describing the objects owing to various requirements or perspectives; however, these metadata concepts should proceed for extensions of specific domain requirements and enable the openness of these object descriptions. Moreover, the openness concept for handling devices and context information is highly associated with the term that is equivalently referred to as the Internet of Things, Internet of Robotic Things, Web of Things, and so on. The 'Things' concept should be first explored from the recent research of Web of Things (WoT). WoT broadly covers the concept of Things that can be deployed and shared on the Internet, and it provides a central building block in the W3C. WoT provides terminologies for specifying detailed aspects, in particular, WoT-Architecture and Thing Description (TD) associates, such as a TD context extension, TD information model, TD processor, TD serialization, or TD document [19–21]. In particular, both WoT and TD consider the terminology of personally identifiable information (PII) [19, 22], which can be used to identify the natural person to whom such information relates or may be directly or indirectly linked to a natural person. Considering the term of Things, in this context, PII also includes not only the profile of persons but also any objects, including devices and datasets that are gathered from it, which can interact with users. Moreover, the concept of PII is fairly considered in ISO-IEC-29100, which is related to information and communication technology (ICT) systems and indicates the definition of objects or components in the real world that may be interchangeable, processing them into the possible context parts. Thus, the extensible interface in the WoT that has descriptions and collectable datasets from the Things needs to be explored for the consideration of context as information sets. There are two major studies for building a knowledge base to extend WoT and TD associates.

First, the study of Sensors, Observations, Actuation, and Sampling (SOSA) ontology [23] provides broad perspectives of sensor devices, observations from it, actuations for processing collected observations, and sampling to represent the value of a property, respectively. It mainly focuses on building meta datasets for Things with four concepts of each SOSA component and representing them as a knowledge base. The SOSA ontology contains not only its metadata but also the concept of abstracting layers that are described in the sampling parts. When building metadata for objects using the concepts of WoT and TD, detailed perspectives and studies are conducted to apply them to industrial fields. Therefore, it is highly desirable to follow the metadata and knowledge parts from the WoT and TD guidelines.

Second, the Semantic Sensor Network (SSN) [21] provides deeper aspects of representing metadata or Things themselves. Using meta datasets from both related works of SOSA ontology and SSN enables detailed perspectives to be gained on the handling of gathered values which can be called fragments of processing context. Handling these values involves various aspects in a view of abstracting context information gathered from the devices, such as composing abstracting levels from the bottom of the deployments, processing raw datasets into well-known measurement units, and representing them as semantic knowledge parts. Janowicz and Compton [24] aimed to provide a simplified ontology design pattern to build a Stimulus-Sensor-Observation concept. The importance of considering values that change simultaneously is highly associated with the context-aware system requirements when the system provides dedicated services to users. These consistent monitoring values should be considered in the QoC-enhanced semantic IoT model, first-class-abstraction, and modeling them [25–28], which indicates and categorizes context information into three types: inferred, sensed, and static. Expressing these concepts enables an improved interpretation when gathered values interact with the

environment, particularly in IoT-related fields. Handling gathered values for visibility and interpretation, processing semantic representation, including aligning them with well-known measurement units [29, 30], and reasoning the datasets with axioms or adding SWRL implementation for meaningful statements [27, 31–33] are required. Although SOSA ontology and SSN provide arbitrary interfaces for the further implementation of the devices, the Things description in context abstraction levels should be considered. When systems handle values or status from sensor devices, especially in context-aware systems, they require information sets such as signals, sensing values, and profiles that should be aligned with equivalent levels or degrees following one of the certain concepts for the users and systems. Thus, it is necessary to focus on the context aspects and its abstraction process [31, 34–37]. That is, it generalizes the processes of data abstraction based on adapting metadata concepts and specifying additional information to describe details of certain levels of context information. Related studies have been conducted on handling signals or sensing values to meet the requirements for data fusion, acquisition, and processing of sensing data domains. One of the models referred to as the Joint Directors of Laboratories (JDL) Data Fusion Model has been maintained as the standard for defense data fusion systems, which has the advantage of collaborating associated intelligence into dissemination [38–41]. The JDL data fusion model is well designed for the flows of perception, data processing, and evaluation of associate intelligence including sensing data. Other studies on the acquisition and processing of sensing data on robotics applications have demonstrated that diverse aspects and objectives exist depending on how robotic devices can provide solutions to issues regarding automation, control, and management [42–44]. These studies have specific procedures or equations for processing the sensing values for further applications. However, there should be certain ways to generalize this procedure, which has not been broadly conducted. Therefore, there is a lack of concern that heterogeneous devices, such as sensors, profiles, and robotic devices, including the JDL data fusion model, have difficulty picking up these phases in a certain manner. Owing to these concerns, individual sensors or entities should be investigated. We refer to this procedure as "data enrichment," which will be introduced in Section 3. The data enrichment procedure provides details on how to use these processed information sets.

Third, there are other considerations when context-aware systems work in collaborative environments of IoT and robotics [45]. One of the main requirements for context-awareness and abstraction is to enable flexible, user-customized, and autonomic services based on the related context of IoT components and/or its users. It also emphasizes that context-based information forms the basis for taking actions in response to the current situation or condition, possibly using information sets of sensors or actuators. The importance of the use of context and providing autonomous services to users in semantic information was also highlighted by Mahieu et al. [46]. Recent developments in robotic devices include profiles of functionalities through ROS environments [47]. The ROS provides a soft abstraction layer for each part of the hardware or logical modules as nodes. Nodes consist of various functionalities in a simple manner, such as node names, topic names, and message types. When software developers attempt to test or operate robotic devices, they usually follow the built-in default settings, although they are not familiar with the ROS environment. To overcome this issue, a variety of data description structures with the JSON API have been established in the study by Rosbridge [48]. This package can reduce the efforts of the developers who are familiar with ROS when testing the functionalities. Conversely, features such as developing interfaces should be provided to non-ROS developers. Once key factors to control robots are provided, releasing interfaces can be solved in a much easier manner. Recent and popular implementations of robotic devices have been shared in open-source projects in an ROS manner [49–52], and functionalities of robotic devices are described in specifications or node configurations compatible with ROS. Therefore, the proposed scheme should contain fields for functionalities, such as function names, function parameters, function return types, and topic or message names for ROS compatibility. Table 1 summarizes the aforementioned related studies for easier comprehension.

## 3. Extensible Data Enrichment Scheme

In this section, an extensible data enrichment scheme referred to as ThingsMetadata is proposed. The goal of the scheme is to provide description sets to service developers who have an interest in handling objects in a real-world scene. The description sets mainly consist of two parts. One is a meta dataset for characteristics of the objects, and the other meta dataset for the collected sensing values to process from the bottom of context perspectives. ThingsMetadata is an XML-based implementation, following the semantic web stack manner and aims to build how both metadata and sensed values can be processed in the literal state and then be represented into triplets utilizing the RDF model. In particular, the proposed scheme integrates context-level stages, including sensed value results, to provide certain statements to context-aware systems. To understand ThingsMetadata, the concept of context information and its definition are also introduced. This section consists of five subsections, starting with the requirements for the scheme through the understanding of the context information in the IoT environment. Subsequently, the degrees of context information level along with ThingsMetadata are described.

*3.1. Overview of the Context Information.* In general, the perspectives of context in the IoT environment include diverse aspects. In this study, we mainly focus on the context information term. Before providing an overview of the context information, Section 2 summarizes the requirements and considerations.

> Heterogeneous devices are deployed in the IoT environment such that descriptions including profiles,

Table 1: Summary of issues discussed regarding the context aspects in related works, Section 2.

| Issue | Key factor | Consideration |
|---|---|---|
| • Supporting openness | • Metadata | • Objects in the real world are nonunified to describe characteristics and implement interfaces for various domains.<br>• Datasets are individually structural, different, or unstructured formats..... |
| | • Knowledgebase | • Metadata concepts provide possible solutions for the openness issues. |
| • Lack of guidelines for handling context as data enrichment | • Behavior how datasets are processed | • Sensing values can be collected and managed in generalized behaviors. |
| | • Building up information sets into equivalent abstraction level | • Each processing phase has independent concepts or equations for measurement units. |
| | • Readability for users and relevant systems | • Processed results should be represented and interpreted in a simplified form to overcome individually different query usages. |
| • Functionalities provision for nonskilled users | • ROS-based functionalities | • ROS-based functionalities consist of combinations of nodes, topics, and messages. |
| | • Interfaces for readability | • Rosbridge is one of open-source projects that interconnect with both ROS and non-ROS operating systems. |
| | • Resolving conflicts for service developers and users | • Relevant surrounding information sets, such as how to control robots or manage services, should be provided as a metadata element. Then, they help to provide a broad view of implementation to the nonskilled users. |

access information, and sensed data types should be formed in unified structures.

Although the descriptions can be formed in complex ways, context information should be presented to users in a simple manner, and the systems can be considered as "easy to manage" for further usage.

Processed data and their descriptions should be handled in a reliable manner to trace each processing step in case of malfunctions or corrections of the descriptions.

To meet these requirements, the definition of context information should include detailed concepts. Therefore, we consider not only the meaning of the ordinary context, as discussed in Sections 2 and 3, but also take a deeper look at the definition of context information. The definition of context information in this paper is described in Definition 1, and the terms or symbols used here are presented in Table 2.

*Definition 1.* Context information refers to a subset of contexts that contains the following three concepts:

Context information follows the same definition as the research by Dey [53] which is as follows "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

Context information consists of abstracting procedures such as building up ThingsMetadata, low-level context, high-level context, and situational information. These procedures provide ways to describe profiles such as the information for accessing sensor devices, types of sensed values, and converting gathered values into URI or literals as vocabularies in a group.

Context information contains a part of the state data or literals for source information and sensor devices in the IoT environment formed in a ThingsMetadata scheme, and each relation is represented as a triplet $\boldsymbol{T}$, where $\boldsymbol{T} = (s,p,o)$ from the RDF data model [18].

Our definition of context information is particularly designed to deal with metadata and their sensed values in IoT environments. Metadata such as profiles, descriptions, and objects in the real world can be considered as static information owing to their characteristics or profiles that do not change. Compared to the typical meta datasets above, sensed values can be repeatedly provided as live feeds or new state information from the deployed environments. It also considers metadata and sensed values simultaneously because they have the same source and should not be dealt with separately. Supposing we obtain information such as temperature at a specific location, regardless of how different types of sensor devices are deployed and assuming their sensed values may be using different types or units of measurement, we determine feelings of bridging temperature and the fixed state (coldness or warmness), even a thermometer can be used to determine the temperature using designated well-known units (i.e.,°C or °F). Finally, both metadata and sensed values should be considered.

While handling sensed values from the raw data to the human and system understandable state, abstracting these procedures should also be considered as data enrichment. Our proposed scheme provides these steps with four degrees of abstraction. The following abstraction steps are modeled in accordance with the human recognition system shown in

Table 2: Terms that are used in the extensible data enrichment and context information levels.

| Symbol | Name | Description |
|---|---|---|
| $s$ | Subject | A vocabulary for representing the thing to be perceived |
| $p$ | Predicate | A vocabulary for the relationship between $s$ and $o$ |
| $o$ | Object | A vocabulary for representing the state or description of $s$ |
| $\boldsymbol{T}$ | Triplet | A combination of $s$, $p$, and $o$ in the form of $< s, p, o >$ |
| $\boldsymbol{G}$ | Group | A set or union of triplet $\boldsymbol{T}$ |
| $\boldsymbol{V}$ | Vocabulary | String literals that can be chosen from the ThingsMetadata description |
| $\boldsymbol{U}$ | URIs | A set of uniform resource identifiers |
| $\boldsymbol{B}$ | Blank nodes | A node in RDF graphs representing an anonymous resource |
| $\boldsymbol{L}$ | Literals | A string or a numerical sequence of letters or digits |
| $F(X)$ | Function for domain X | A function that defines how to process data enrichment |
| $X$ | Domain X | A set of input data for data enrichment functions |

Figure 1 as a generalized way of context abstraction that how the perceptions can be made in such levels, including profiles of robotic things, for other purposes.

Figure 1 shows the overall concept of the proposed data enrichment method. The data enrichment method aims to plan the levels of processing context information in IoT and Robotics environments, as discussed in Section 2. Each step provides a way of describing context information based on the degrees from the sensed values to the human-understandable information. These steps involve data enrichment and data abstraction models. One of the requirements that we explored was the "easy to use" context. Our data enrichment scheme provides steps for knowledge, from making profiles to converting sensor data by utilizing our model. Through the data abstraction model, the context information sets were finally represented in a triplet form. This triplet form provides a simple and powerful description. All input datasets are described in the ThingsMetadata format, which provides the following advantages:

*Interpretability*. Every data element can be represented in a triplet form, which can easily provide readability to users and service developers. The datasets described in this triplet form can also be processed by any SPARQL-relevant applications. These applications are also considered as knowledge base, and ontology tools and interfaces are also provided.

*Adaptability*. The ThingsMetadata form provides various ways and data fields for objects in the real world to describe their profiles, sensor data types, processing sensed data, and so on. Previous studies have similarly considered these profiles [14–18]. Furthermore, our ThingsMetadata has sensor data types that can be converted into meaningful states.

*Extensibility*. The resulting datasets are followed by the ThingsMetadata structure, and they are represented in the RDF form for ontology. Other elements such as static metadata sets, which are explored in Section 2, can additionally be described depending on requirements and service domains. In this study, our abstraction model only considers sensor devices, user profiles, and robotic devices; however, other objects in different domains can be attached for further considerations.

*3.2. ThingsMetadata.* ThingsMetadata as the first degree of data enrichment is a scheme or any instance following the structure that contains descriptions for the user profile, sensor devices, or robotic devices. This is a new term for objects in real-world scenarios.

*Definition 2. ThingsMetadata.* A subset of context information that refers to a description for any object that has attributes under specific named elements. ThingsMetadata is the first stage of a context abstraction model that creates higher context levels, such as low-level context, high-level context, and situational relationships. The concepts and details for ThingsMetadata are described as follows:

ThingsMetadata, shown in Figures 2 and 3, can be described in two different data formats including, but not limited to, XML, JSON, or any structured documents. Figure 2 shows the first part of the entire ThingsMetadata scheme as shown in static information sets. In Figure 2, examples of metadata are placed; however, metadata sets can be replaced by the WoT guidelines as discussed in Section 2. Figure 3 shows the rest of the scheme, that is, the active information sets that are base information for data abstraction in this study. ThingsMetadata contains general profiles categorized in a description, an entity name for their ID, interval, data format, and transform. A primary key for each ThingsMetadata is an entity name, which is described as _id.

In general, ThingsMetadata provides both static and active information. Static information refers to unique characteristic information, such as profiles, data types, or converting rules for sensed values. Active information refers to any gathered or converted data from sensed values as time flows. The datasets of the active information are described using the transform element. The details of the active information are described in the following sections.

To obtain sensing values, fields under Access should be filled with connecting information, such as plain text and private credentials. Although profiles mostly contain string literals, entities for access follow the rules listed in Table 3.

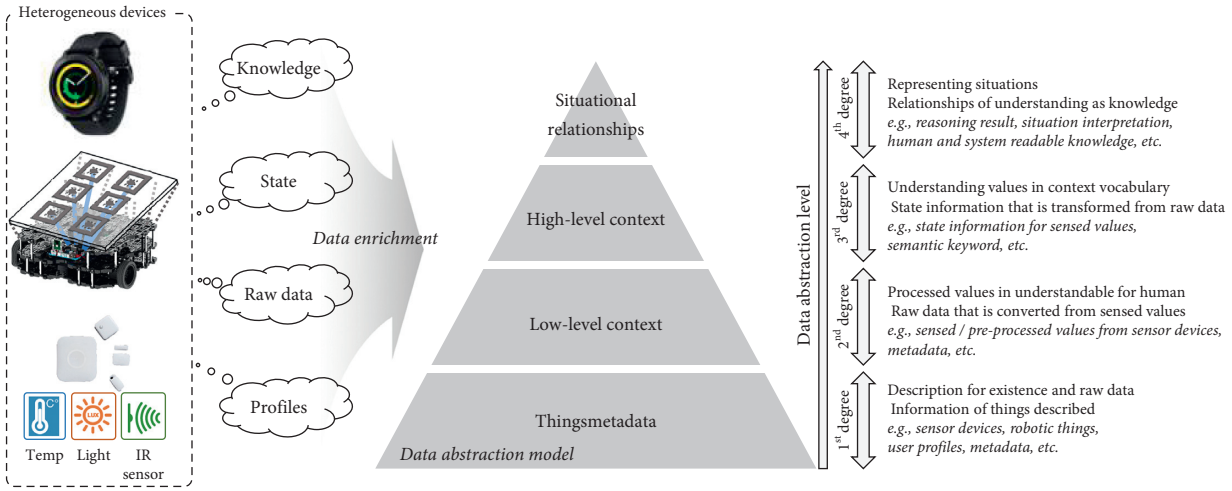



Figure 1: Overview of a data enrichment concept considering metadata and sensed values with four degrees of abstraction.

Fields under Common, Sensor, or Robot in description are used to describe properties for logical mappings through ontology. Then, users or service developers can find profile information from the desired input string sets or literals as static information set with entity.

Fields under DataInfo are used to bind logical data fields or entities. Properties under this field are used to first find references from the given entity names, not as literal. Then, the users or service developer can describe the combined status from multiple devices or sensing values.

Fields under DataFormat provide data type interfaces for heterogeneous devices connected by connectionType under the description information. By filling the types of sensed data, this procedure avoids type mismatches during a gathering data phase in runtime during the collection of raw datasets. Raw datasets will be collected by valueType profile and Access in Description as incoming channels.

Raw datasets refer to values without well-known or human-understandable units; for example, the temperature degree can be obtained from sensing values upon the resistances in the sensor devices. The temperature can be calculated from the sensing values using formulas, depending on the devices used. In this case, the sensing values are raw datasets, and the temperatures are abstracted values with a rule that converts the raw datasets into temperatures.

Raw datasets can also be processed into meaningful information for users or systems using data enrichment or data abstraction procedures. These procedures are performed by the descriptions under a Transform element that contains detailed procedure steps. Users and systems can use either type of data bindings: raw data or one of the further procedures that uses Transform description. In this paper, one example of using situational relationships is introduced; however, the rest of the data bindings can be found in the same way with the ThingsMetadata usage.

*3.3. Low-Level Context.* ThingsMetadata provides descriptions of connection methods or access information to target objects in the real world. Although properties are described as a profile of the ThingsMetadata description, there are issues that need to be solved, such as mismatching measurement units. For example, we assume that there are two different types of sensor devices; one is a well-designed sensor device that provides temperature values in degrees Celsius. The other sensor device is a manually assembled heterogeneous sensing device with software modules that are programmed by open sources and are mostly representing raw values in electrical observations. Both provide values for temperature; however, the latter provides raw sensing values. In this case, the latter one should be converted into the same measurement units as what the first device provides before users or service developers can understand them. In a low-level context, the aforementioned examples can be processed into the same measurement unit by applying such functions that are described manually. Then, service developers can bring adjusted sensing values and their perspectives. Figure 3 shows the remaining structure of the ThingsMetadata scheme that is used for abstracting sensing values, such as low-level context, high-level context, and situational relationships. The low-level context is the second degree used to process the sensing values in data enrichment.

*Definition 3. Low-Level Context.* This refers to datasets that are converted from raw datasets to values containing human-readable or well-known units. A low-level context can be used when the raw datasets are less sufficient to be represented for the users or service developers to understand the values. The detailed concepts for the low-level context are as follows:

When processing the raw datasets into a low-level context, the procedure itself is called "converting." The converting procedure is performed using the descriptions in ThingsMetadata. The applied descriptions,

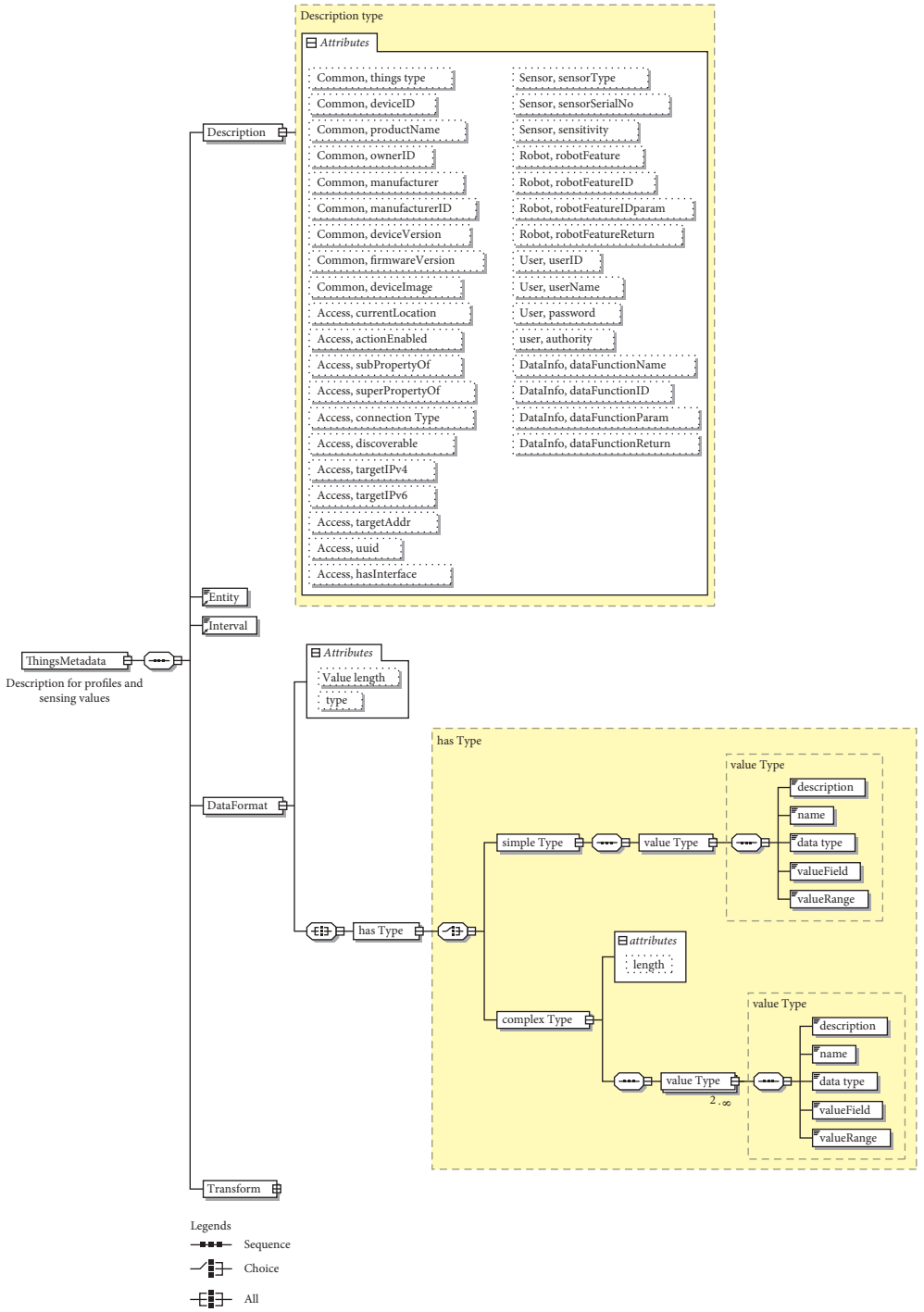



Figure 2: The ThingsMetadata scheme provides fields for profiles and data types of sensing values to connect to real-world objects in the system using higher-level context information. Profiles can provide five types of datasets: common for describing things in the real world, access information to acquire live feeds from objects, sensor devices, user information, and data information for binding logical entities that are already described and be used again for different purposes.

such as data types and conversion rules, were used. However, the collected datasets do not require when they are done with processing to certain measurements, and then this conversion procedure can be skipped.

Data types for origin sources (raw datasets) are described in a HasType element under valueType, which is the same content as the DataFormat from the ThingsMetadata section. The other data types for a target low-level context piece are described in the HasType element under basisValue. The two different HasType and DataFormat elements have the same structures for formatting data types.

The DataFormat and HasType elements have two different subelements: simpleType and

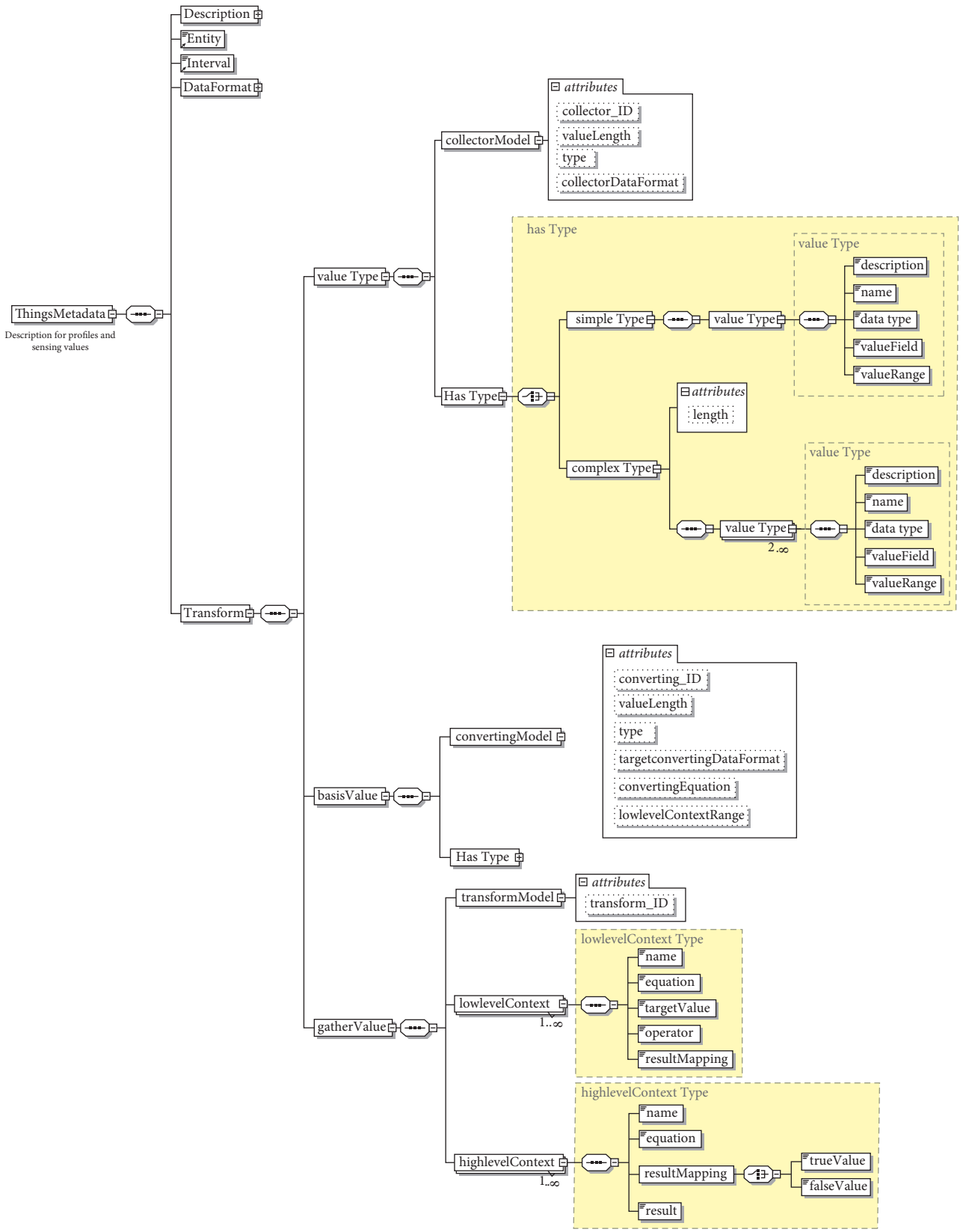



Figure 3: Transform element in the ThingsMetadata scheme comprising of three subelements: valueType, basisValue, and gatherValue. These subelements are used for describing the procedure of abstracting sensor values from raw data to situational relationships.

Table 3: Details for access information entities in ThingsMetadata.

| Entity name | Description | Type |
|---|---|---|
| currentLocation | Name of location for target object | Literals |
| actionEnabled | Whether access fields are on or not | Boolean |
| subPropertyOf | Name of subproperty | Literals |
| superPropertyOf | Name of superproperty | Literals |
| connectionType | Connection type (*bluetooth, BLE, IPv4, IPv6, SERIAL, OTHERS*) | Enum literals |
| Discoverable | Whether the entity is discovered or not | Boolean |
| targetIPv4 | Target IPv4 address | sockaddr* |
| targetIPv6 | Target IPv6 address | sockaddr_in6* |
| targetAddr | Target address (manual) | Literals |
| Uuid | UUID strings | Literals |

*indicates that the following data types have the same structure as POSIX libraries.

complexType. They are both used to describe data types; however, the usages are different by the behavior or correspondence of incoming datasets. While simpleType is used for one data stream channel that provides the output with a single data type, complexType is used for multiple data stream channels containing multiple simpleType sets in a sequence.

Converting rules are described in a convertingModel element under basisValue as attributes. The converting rules are represented as functions and domains, denoted as $f$ and $X$, respectively, for example, $f_i(X_m)$ or $f_j(X_1, X_2, \ldots, X_n)$. All $f_i$ can be defined by the users or service developers to targetconvertingDataFormat, convertingEquation, and lowlevelContextRange for an output data type, the equation $f_i$, and a range of output data, respectively. All domains $X_n$ are the same variables or literals from valueField under HasType.

The results for the conversion procedure are assigned to resultMapping and the remaining elements are put under lowlevelContext of gatherValue if the output data type is correctly cast and their datasets are in the proper boundary according to the range.

The remaining elements can work with the two behaviors to be filled. First, if the elements are empty, the values can be used from the convertingModel. Second, if the elements are filled, the values are evaluated against the convertingModel. Both behaviors can proceed at the validation stage for ThingsMetadata or during the runtime when resultMapping is assigned. Finally, the converted datasets are represented in human-readable and well-known units.

*3.4. High-Level Context.* We defined several parts of context information such as the ThingsMetadata scheme, raw datasets, and low-level context. Further, raw datasets can be processed in two ways (Algorithm 1).

First, they are processed into a low-level context, which represents values with human-readable and well-known units when they have no information about the unit of measurements. Second, raw datasets are in a low-level context, as they are represented by units. A high-level context aims to transform a low-level context, which contains numerical values, into string literals representing their state, including literals defined by the users. In this stage, users and service developers can determine the state of the given or collected values. A high-level context is a third degree to process the sensing values in data enrichment.

*Definition 4. High-Level Context.* This refers to datasets that are transformed from a low-level context. The high-level context can be symbolized as string literal forms representing the states of the target sources. Users or service developers can describe and understand the state in a string literal form, even if the values are aligned with the units when they are at the stage of processing a low-level context. The detailed concepts for the high-level context are as follows:

When the low-level context is processed into a high-level context, the procedure itself is referred to as "transforming." The transforming procedure is performed by using the descriptions in ThingsMetadata. Transforming a low-level context into a high-level context aims to represent their state as literals by the users or service developers that deal with the numerical values collected from the devices or robotic things.

The input data for the transforming procedure are the resultMapping values under lowlevelContext. These input data are assigned to an equation value under highlevelContext. Then, the output data values are evaluated to check whether the result of the equation values is satisfied by trueValue or falseValue, which shows binary outputs.

Transforming rules are described in an equation element under the highlevelContext. The transforming rules are represented as functions $f$ and domains $X$, such as $f_i$ $(X_m)$ or $f_j(X_1, X_2, \ldots, X_n)$. All $f_i$ can be defined by the users or service developers to equation, trueValue, and falseValue under resultMapping.

The results for the converting procedures are assigned to HasType under basisValue if the output data type is correctly cast and their datasets are in the proper boundary according to the range. The converted

*Let* $\boldsymbol{G}$ be a finite group and $\boldsymbol{T}$ = *(s,p,o)* be a triplet from the RDF data model, which is a subset of $\boldsymbol{G}$.
*Define* $\boldsymbol{V}$ = {s,p,o $|\boldsymbol{T}$ = *(s,p,o)* $\in \boldsymbol{UB} \times \boldsymbol{U} \times \boldsymbol{UBL}$, where $\boldsymbol{T} \in \boldsymbol{G}$}
Consider each ThingsMetadata document as an input.
Consider a particular triplet $\boldsymbol{T_z}$ = (*"ThingsMetadata"*, *ofA*, *the assigned literal of "Entity"*) as a root triplet denoted as $\boldsymbol{T_z}$ = ($s_z$, $p_z$, $o_z$).
Consider $\boldsymbol{T_y} \in \boldsymbol{G_y}$, where $\boldsymbol{T_y}$ represents triplets of all adjacent individual and their predicates.
***Generate*** $\boldsymbol{T_x}$ into $\boldsymbol{G_x}$, where $\boldsymbol{T_x}$ contains both $\boldsymbol{T_{y\text{-}i}}$ and $\boldsymbol{T_{y\text{-}j}}$ when subject($\boldsymbol{T_{y\text{-}j}}$) = = object($\boldsymbol{T_{y\text{-}i}}$) after applying the transition rule with chain-reductions.
***Iterate*** all triplets $\boldsymbol{T_x}$ ***as*** $\boldsymbol{Tx}$ = $\boldsymbol{(s_x, p_x, o_x)}$ in the given set $\boldsymbol{G_x}$.
  ***if*** (equalsTo(getSubject($\boldsymbol{T_{y\text{-}j}}$) = = getObject($\boldsymbol{T_{y\text{-}i}}$) ) = = result)
    Make a copy of the $\boldsymbol{T_{y\text{-}i}}$, denoted by $\boldsymbol{T'_x}$ onto $\boldsymbol{G_x}$, and set the predicate and object of $\boldsymbol{T'_x}$ as *isNow* and the result of getObject($\boldsymbol{T_{y\text{-}j}}$), respectively.
  ***else if*** (!( equalsTo(getSubject($\boldsymbol{T_{y\text{-}j}}$) = = getObject($\boldsymbol{T_{y\text{-}i}}$) ) = = result)
      && (IsLiteral(getObject($\boldsymbol{T_{y\text{-}j}}$)) = = true)
    Make a copy of the $\boldsymbol{T_{y\text{-}i}}$, denoted by $\boldsymbol{T'_x}$ onto $\boldsymbol{G_x}$, set the predicate of the $\boldsymbol{T'_x}$ as *has* < getSubject($\boldsymbol{T_{y\text{-}i}}$)>
  else
    Set the predicate of the $\boldsymbol{T'_x}$ as *hasA*
  ***merge*** $\boldsymbol{G'}$ into $\boldsymbol{G}$
end
      result indicates a key name itself in ThingsMetadata structure.
      *$\boldsymbol{T_m}$***'s object** $o_i$ contains user-defined literals.

Algorithm 1: A procedure for building reduced *isNow* or *has<key>* triplets with an SWRL rule.

datasets are then represented as human-readable and well-known units.

*3.5. Situational Relationships.* Once low- and high-level contexts are processed successfully, each entity will be filled with values or string literals of information, followed by the ThingsMetadata scheme. Then, the string literals in the scheme can be used or illustrated to represent their status as keywords in the RDF model. However, there are no rules for conducting and combining keywords to be used as context information. Furthermore, ambiguity certainly occurs in any situation where service developers or users describe scenarios including context information sets, as there are many string literals in the scheme, and they are coupled in a complex structure with different depths. In this subsection, we suggest situational relationships, the final degree of data enrichment.

*Definition 5. Situational Relationships.* This refers to datasets described in a triplet form. Combining the forms of keywords derived from high-level contexts has several options. Using situational relationships with concatenating keywords that are similar to using a predicate between them by rules is recommended. The purpose is to connect descriptions of metadata and sensed values related to objects in the real world directly based on first-order logic. The meaning of first-order logic in this section can be described using a plain statement derived from reducing several logical statements. The detailed concepts for situational relationships are described as follows:

When processing the high-level context, the procedure itself is referred to as "inferencing." The inferencing procedure is performed using the following concepts from semantic web techniques: first-order logic, derivation with chained reduction, and reasoning methods. Situational relationships can be represented in higher-order logic; however, it is mostly focused on building up with first-order logic. The definition of the first-order logic follows relevant research, including the use of quantified variables over nonlogical objects and the concatenation of sentences that contain variables. One triplet example for representing situational relationships based on first-order logic can be expressed as follows: <*Turtlebot1_MeetingRoomA*, *robotFeature*, *Tea_Service3*> or < *Subject*, *Predicate*, *Object* > triplet form as same as the mentioned triplet of (*s*, *p*, *o*).

Keywords in the triplet form should be applied values or string literals from the ThingsMetadata structure. *Subject* can be selected as any entity literal or any object in the real world. Predicate can be any element name from the structure. Meanwhile, *Object* can be one of the values or string literals that represents the state or description of a subject followed by the same *Predicate.*

Input data are the ones of ThingsMetadata entities and their contents. Then, users and service developers can describe triplets as outputs of situational relationships after the filling instance with the structure is done.

There are two types of triplets that can be composed with the ThingsMetadata scheme.

First, there are simple triplets without derivations or chained reductions. The meaning of no derivations is that users, service developers, or systems can conveniently look up values from the triplets.

Second, there are reduced triplets with derivations or chained reductions. The reason for applying the derivation or chain reduction is to make triplets' readable for the users and systems. The reduced triplets can be

developed using an SWRL rule with a blank node B, which is supported by one of the RDF models on semantic web techniques. The procedure for building the reduced triplets is described in Algorithm 1.

When the target service is executed, more than a single observed triplet that reflects the surroundings around users, particularly the sensed value state or description literals, will be checked by the context-aware system. These state literals are represented as a triplet in a reduced form using Algorithm 1. Through Algorithm 1, the user-defined literal state can be processed as generating triplets and be merged into the given triplet group.

Both simple and reduced triplets can be used to describe the statements of the surroundings. In this study, built-in predicates called *isNow* and *has<Key>* are provided as examples to improve the readability and accessibility of the triplets. The details of the two built-in predicates are as follows:

*isNow* is the first built-in predicate suggested in this paper, which processes sensor datasets such as low-level context, high-level context, and situational relationships. In particular, the predicate of *isNow* interconnects between the entity's name and target sensor data literals. By default, *isNow* reduces statements in a high-level context. For example, resultMapping, which describes the state of the transformed value in highlevelContext, can have multiple triplets in several transitions. However, these triplets can be represented in a reduced way when *isNow* is used as a predicate for each entity and describes the state information gathered from the sensors.

*has<key>* is the second built-in predicate suggested in this study, which improves the readability of triplets. *<key>* can be any element name from the ThingsMetadata scheme. For the nested elements, they are separated by dots at different depths and *has<key>* provides better readability. For example, *<Turtlebot1_ConferenceRoomA*, *robotFeature*, *Tea_Serv1>*, the example triplet, can also be described as *< Turtlebot1_ConferenceRoomA*, *hasrobotFeature*, *Tea_Serv1>*. For the scenario description, the *hasrobotFeature* predicate can be accessed with *Description.Robot.robotFeature*.

By applying data enrichment steps with the ThingsMetadata scheme described in this paper, the surroundings can be described in a quantitative, measurable, and understandable way for the users and systems. In particular, the users can take advantage of their interpretations because the triplets, including user-defined ones, can be delivered in a readable manner. Furthermore, services can be operated by checking and verifying these information sets in a unified representation.

## 4. Experiment

In this section, an experiment related to ThingsMetadata is presented. This section comprises three sections. First, an overview of the experiment is presented which includes a situation that we demonstrate in plain text, along with a summary of the devices used in the experiment. Second, a subsystem called the context-aware workflow language (CAWL) engine is introduced as a middleware application that manages surroundings and service activations. Third, the procedures of data enrichment were examined using the ThingsMetadata scheme. Briefs of situational relationships are also presented to provide better explanations.

*4.1. Experiment Overview.* We demonstrate how the data enrichment scheme works and can be applied to practical scenarios. Descriptions of the scenes are explained below.

UserA works at the office as a general manager. One of the job responsibilities of UserA is to arrange meetups at one of the conference rooms. Today, UserA prepares to serve tea for the participants. There are two devices that UserA can control in the conference room. One is a wearable device (Samsung Galaxy Gear series), and the other is a robot (Turtlebot series) on the table in the conference room. UserA wants to control the service named serving tea through the robot. And this service will be activated through the wearable device by UserA.

Details of the plain text description are shown in Figure 4; both devices should be prepared for the scenario. First, the robot (hereinafter Turtlebot) can run as a node on the ROS and should be assembled with various components, such as sensors, actuators, frames, and wheels. Second, the wearable device (referred to as Gear) can interact with the user through Tizen-based user applications. In this study, to provide tea services to the users considering minimized requirements, Turtlebot is assembled with mobile parts, infrared range sensors, and frames for cup slots. The Gear provides a native app that interacts with users using a button on the screen display. Figure 5 and Table 4 show a blueprint of the service scenario and the summary and details of the two devices described in the ThingsMetadata scheme, respectively. There were two purposes of the test scenario. First, to see how the users interacted with IoT devices and robots along with context abstraction levels. Second, seeing whether the capabilities of service descriptions can be broadly applied when robots can solely utilize service invocations collaborating with user interactions or gathered values that are changed, including arbitrary implementation using the serving plate. The serving plate can bridge different context levels, from ThingsMetadata to high-level context information. Then, the CAWL engine proceeds to look up the situational relationships that represent the states of the specific serving moment.

A profile summary and description of the devices used are presented in Table 4. *Turtlebot1_ConferenceRoomA* and sensor devices (*IR sensors* and *serving plate*) are combined into a prototype, and parts of the ThingsMetadata description are shown in Figures 6 and 7, respectively. This assembled device provides a tea service by the CAWL engine introduced in Section 4.2. Controlling functionalities using this prototype can be done manually, and we demonstrate one of the options for working with the CAWL engine in this

UserA works at the office as a general manager. One of the job responsibilities of UserA is to arrange meetups at one of the conference rooms. Today, UserA prepares to serve tea for the participants. There are two devices that UserA can control in the conference room. One is wearable device (Samsung Galaxy Gear series), and the other is a robot (Turtlebot series) on the a table in the conference room. UserA wants to control the service named serving tea through the robot. And this service will be activated through the wearable device by UserA.

Figure 4: Explanation of tea service for the conference room.

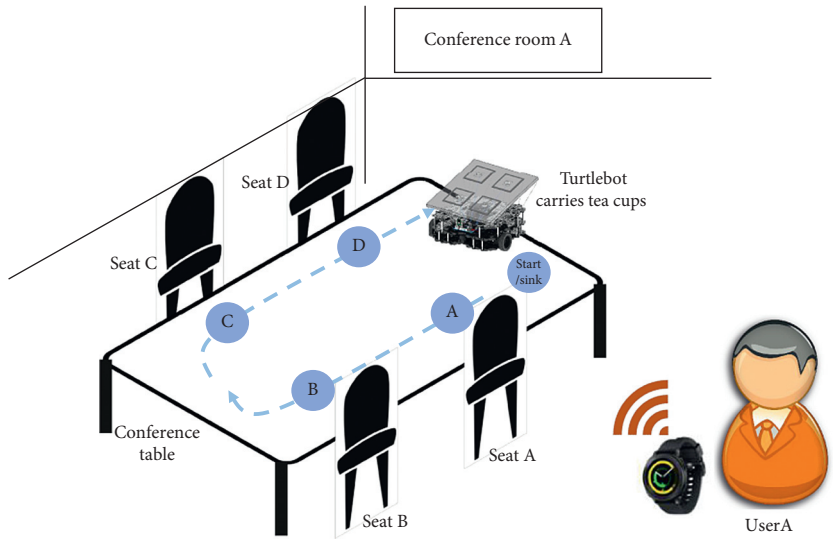



Figure 5: Blueprint of service scenario for serving tea. UserA activates the service through Gear, with an application installed on it to send signals to the system for data enrichment. Once service is activated, Turtlebot will move from the starting point to each seat spot and will wait for the guest to pick up the cup. The service scenario is controlled and managed by the CAWL engine, which is explored in Section 4.2, and the results of data enrichment will be used for checking the surroundings as represented in a triplet form.

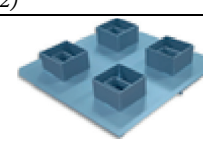
experiment. The CAWL engine provides a way to describe the surroundings including the user interactions in a triplet form with scenario documents to validate whether they are confirmed as choices of true or false states. Finally, it activates services to the devices. We intend to plan and describe the scenario documents for the service invocation. Service developers can find how to implement data enrichment procedures steps such as accessing devices, collecting raw sensing values, converting them to the level that service developers can understand, and composing them into triplet-based representations that are used to invoke service applications based on the proposed scheme.

Initially, we checked whether **Gear** interacts with UserA. Once UserA interacts with Gear, the tea-serving scenario begins. **Turtlebot** starts moving from a home position, visiting seats #1 to #4 and checking whether each cup on the serving plate is taken or not. During the process of checking the status of the cup, IR sensors generate sensing values that are considered as raw datasets and are available to proceed with the data enrichment procedure with ThingsMetadata. Further details and usages are provided in Section 4.2.

*4.2. Context-Aware Workflow Engine.* CAWL [54] is an XML-based language that enables the planning of business processes and service automation in scenario documents. It is also a branch of workflow languages, especially in distributed systems or ubiquitous environments. The CAWL engine is a combination of implemented middleware modules, for example, parsing components in the documents, transmitting next workflows whether components are matched to the current state or not, and activating functionalities. The scenario document contains description sets of surroundings, which are described in a triplet form. The phases of the document can be planned in several ways. Figure 8 shows the scenario when guests are ready to take each cup, while **Turtlebot** is on the move.

*4.3. Data Enrichment Procedure.* As Figures 4 and 5 illustrate the blueprint of the experiment with procedures performed by the CAWL engine, in this subsection, we demonstrate how data enrichment works with ThingsMetadata. We have implemented submodules for processing the ThingsMetadata scheme and instances, such as a ThingsMetadata parser, device interface, RDF converter, SWRL module, and ontology loader. By using these modules, each entity based on ThingsMetadata can be processed by identifying the devices in the data enrichment procedure. Finally, every dataset of situational relationships from the entities can be generated. Figure 9 displays the brief skeleton of one IR sensor

Table 4: Details for access information entities in ThingsMetadata.

| | | | | |
|---|---|---|---|---|
| Product figure |  |  | | |
| Product name | Samsung galaxy gear 2 | Turtlebot 3 | | |
| Entity name | Gear_ConferenceRoomA | Turtlebot1_ConferenceRoomA | | |
| Device type | Access, sensor, user | Access, robot | | |
| Manufacturer | Samsung electronics | Robotis | | |
| Owner | UserA | UserA | | |
| Access method | IPv4 | IPv4 | | |
| Access address | 192.168.15.10 | 192.168.15.152 | | |
| Service name | Service_Touch1 | Echo_State/Position_State/Robot_Move | | |
| Service param | N/A (user interaction) | Null/Null/struct coordinates | | |
| Service return type | Integer | Integer/Integer/Integer | | |
| Data enrichment | Low-level context to situational relationships | | | |
| Transforming rules | • true: RUN_TeaServ1<br>• FALSE: STOP_TeaServ1 | Each column prints one of states below<br>*State_READY(20)*<br>*State_TOHOME(21)*<br>*State_MOVING(22)* | *Position_HOME(10)*<br>*Posision_Seat#(#)* | *Change states* |
| Product figure |  |  | | |
| Product name | IR proximity sensor -sharp GP2Y0A41SK0F | Serving plate prototype | | |
| Entity name | Robot_Turtlebot_IRSensor# (# indicates number) | Robot_Turtlebot_Serving | | |
| Device type | Access, sensor | Access, DataInfo | | |
| Manufacturer | Sharp | N/A | | |
| Owner | UserA | UserA | | |
| Access method | Others | IPv4 | | |
| Access address | PIN addr | 192.168.0.103 | | |
| Service name | IRSensor_State | Serving_State | | |
| Service param | Null | 4 integer values from *Robot_Turtlebot_IRSensor#* | | |
| Service return type | Integer | Integer | | |
| Data enrichment | Raw to situational relationships | Low-level context to situational relationships | | |
| Converting/ Transforming rules | *Converting rules 1309/((RawData/4) - 3)*<br>*cmTransforming rules*<br>• INIT: *CUP_EMPTY*<br>• true: $Result_{convert} > 5$ =>*CUP_TAKEN*<br>• FALSE: *CUP_READY* | *Transforming rules*<br>• INIT: *SERVING_EMPTY*<br>• true: $CUP\_TAKEN_i > 0$, then, *SERVING_TAKEN*<br>• FALSE: *SERVING_READY* | | |
| Notes | Four sensors are attached into *Robot_Turtlebot_Serving* | Manually assembled, a control motherboard are attached into *Turtlebot_ConferenceRoomA* | | |

description with the serving plate and how to proceed with data abstraction using with Transform description to form situational relationships (at the moment of C10, Figure 8) that are reduced and generated following the rules presented in Table 4 in Section 4.1; Figure 10 demonstrates the tea serving service. In Figure 8, it is shown that the scenario document checks such states to proceed with each step of servings. At this point, additional conditions can be described in the scenario document when circumstances are changed.

## 5. Discussion

In this study, we investigated the interconnection of objects in real-world scenes that can provide services in the robotics domain by handling a data abstraction procedure that is carried out by the proposed scheme referred to as Things-Metadata. By applying this scheme, there are considerations for integrating objects such as robotic devices, individual sensor devices, and any object that can be deployed around users.

In general, software engineers encounter difficulties in integrating deployed objects in the real world and handling sensing values in various formats because they can be gathered from different software and/or hardware interfaces or by using connection methods. This difficulty can be solved by showing information sets of endpoints, such as robot service names, function call routines, and parameters with data types, as well as Swagger, an open-source project that

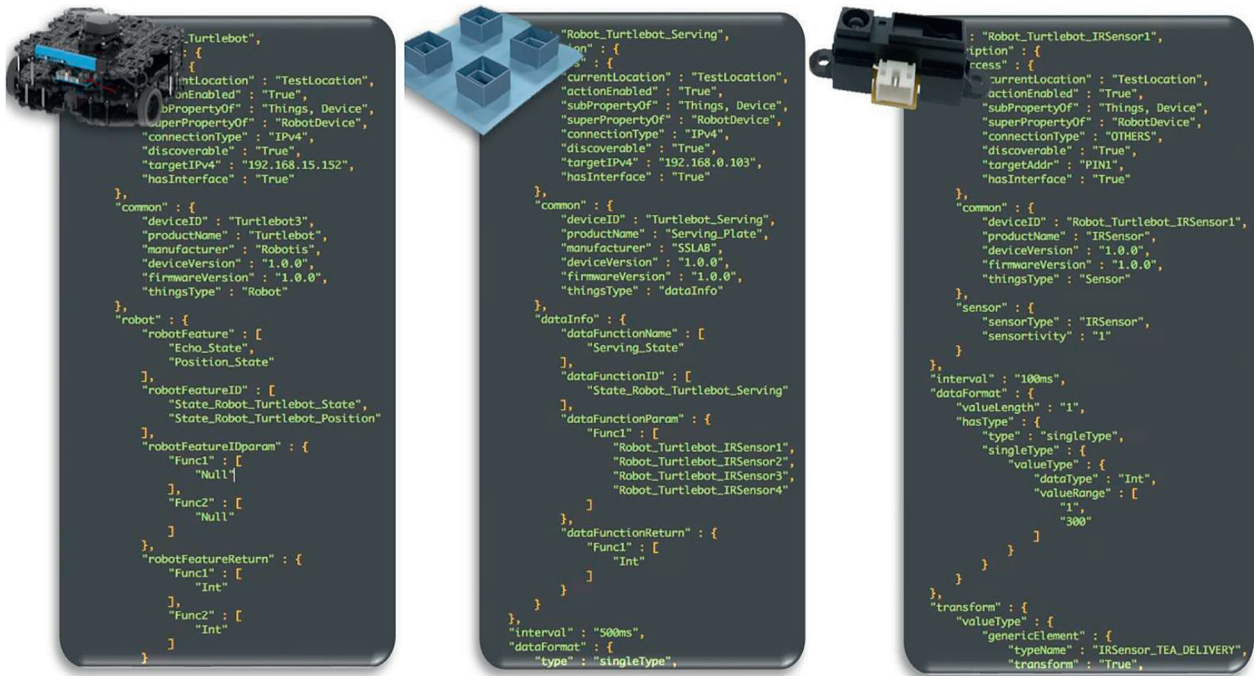



Figure 6: Part of ThingsMetadata description for the devices related to Turtlebot1_ConferenceRoomA.

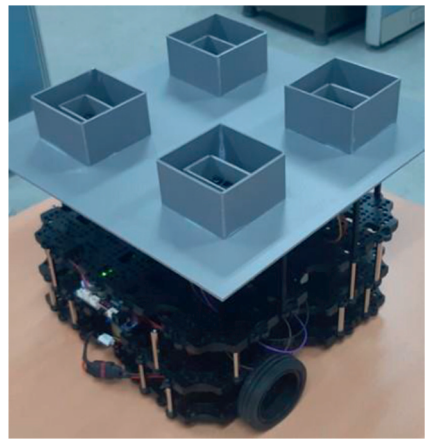


Figure 7: Prototype Turtlebot, combined with turtlebot3, IR sensors, and serving plate.

also provides metadata for RESTful web service APIs [55]. Therefore, nonexpert developers in areas related to the robotics domains can find references to deploy robot services with the proposed scheme. Relevant studies that have been presented in Section 2 can provide broad perspectives on handling datasets from the bottom of setting up devices; however, context information sets must be covered as they are used as base information or input datasets for complex service invocations.

The proposed ThingsMetadata scheme shown in Figures 2 and 3 provides information on how datasets can be collected and handled in different aspects of abstraction levels. Meanwhile, software engineers or service developers can feedback the context information in the middle of data abstractions. The advantage of processing datasets is that both transparency and openness can be achieved in structured formats. It also provides another benefit of interpreting understandable datasets to users who may be interested in data transformation details.

For context information usage, these datasets should be processed in a visible manner. In the Experiment section, the data abstraction procedure is presented in the style of a structured scheme and description logic (DL) graph to visualize the details. In semantic web domains, advanced context-aware systems need to determine information sets that should be highly relevant, understandable, and accurate for both human and computer systems. Typically, approaches using ontology are applied, and the results from the experiment can be used to represent states for interpretation.

The aspects mentioned above are discussed to proceed with data abstraction. Although ThingsMetadata

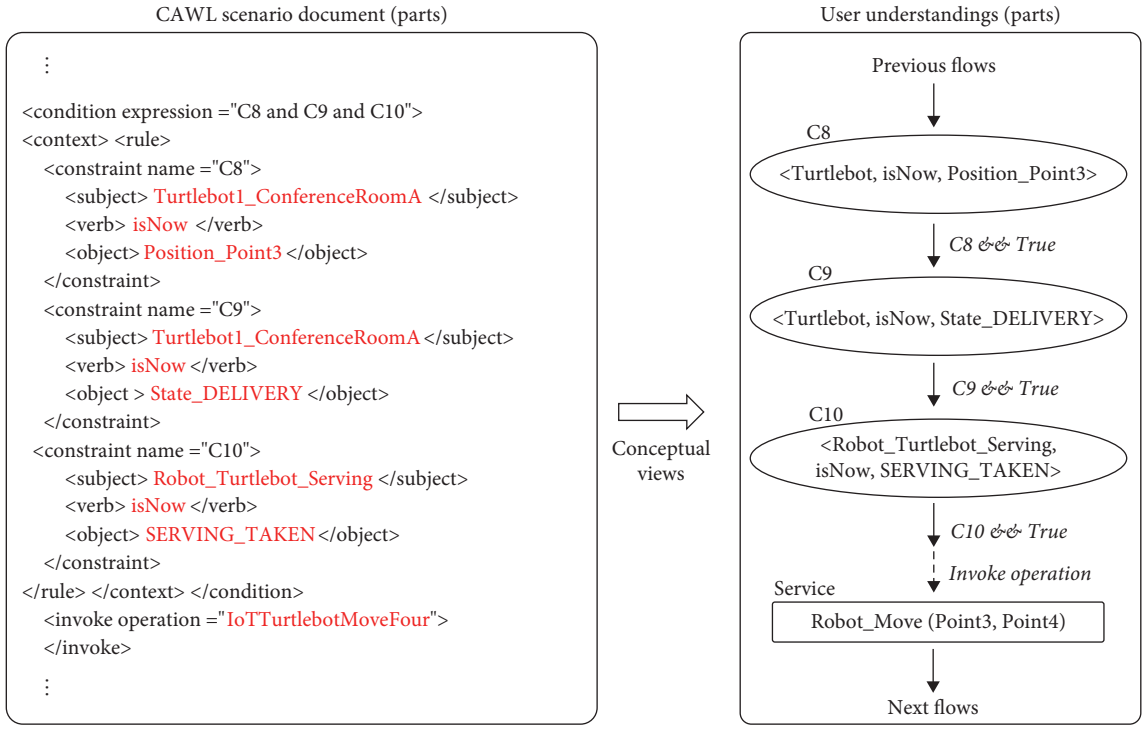



Figure 8: Example of the CAWL scenario document and a conceptual view of triplets processing.

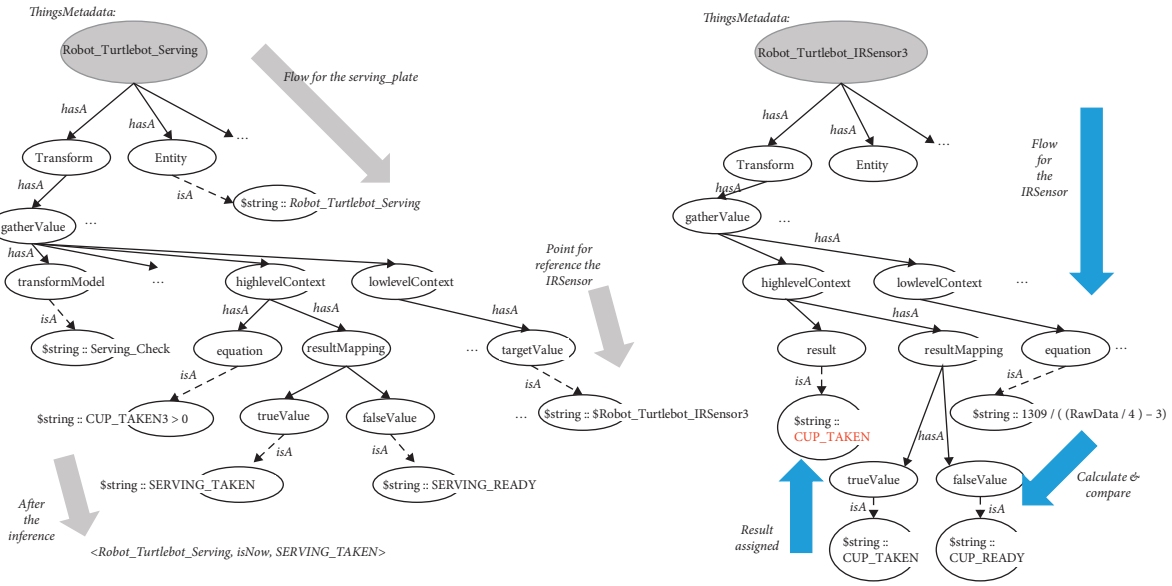
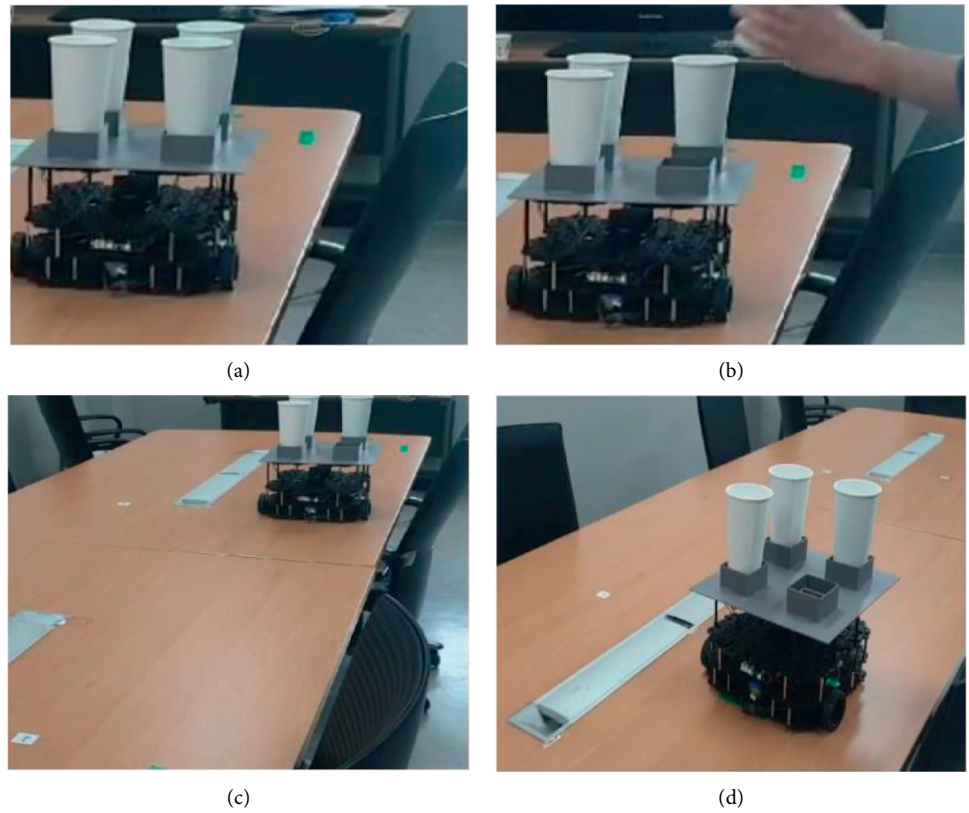



Figure 9: Brief description logic (DL) graph of the data enrichment procedure based on the ThingsMetadata scheme for the IRSensor and Serving plate. The arrow order (blue and gray) does not matter whether trueValue or falseValue is selected.

focuses on integration with various sensor device types, the capability of handling objects and datasets should be performed as recent datasets become more complex. ThingsMetadata currently handles discrete datasets such as single or complex types of sensing values within the entity-wise scale when the data acquisition phase is processed. Recent modern systems mostly collaborate with continuous datasets that are stored in NoSQL-based databases for use in machine learning or deep learning analysis applications. Therefore, the ThingsMetadata scheme should cover other factors that can be used in the analysis processes for future work.



(a) (b) (c) (d)

Figure 10: Demonstration of the tea serving service as mentioned and explained in Experiment section. As infrared (IR) sensors keep checking whether the users take individual cups or not, **Turtlebot** keeps listening to the service invocations from the collaboration with CAWL engine using the data enrichment procedures. (a) Ready for user response. (b) User interacts with Turtlebot to pick up one cup that is positioned on the serving plate. (c) Turtlebot moves to the next position that is indicated in a service document such as in Figure 8. (d) As **Turtlebot** finishes moving, it becomes a ready state as in (a).

## 6. Conclusions

In this study, an extensible data enrichment scheme, ThingsMetadata, has been proposed to address the issue of openness, to process values from heterogeneous sensor devices, and to control functionalities for data enrichment. When collaborating with sensor devices through the proposed scheme, context-aware systems can handle various complex data types of sensing values. In the Experiment section, we presented the data enrichment flows, which are described in four levels of data abstraction layers. The advantage of the proposed scheme is that it can provide both users and developers transparency, openness, and broad adaptability to the devices and sensing values. One of its contributions is to identify context concepts that improve interpretations of collectable datasets in IoT environments on the semantic web stack. Future work is required in terms of applying the ThingsMetadata concepts in edge-computing environments that are managed in individually isolated and dedicated demands with lightweight operating system kernels. Further extensions include improvements in usability and the development of service domains. Hence, it is necessary to focus on editing features that collaborate with relevant parsers and generators. Moreover, the coverage of analysis procedures regarding the ThingsMetadata scheme for emerging analysis results in context information should be determined.

## Data Availability

All the data along with the scheme and description sets during this study are included within this article.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was supported by Basic Science Research Program (no. NRF-2019R1A2C1007861) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT).

## References

[1] A. Whitmore, A. Agarwal, and L. Da Xu, "The Internet of Things-A survey of topics and trends," *Information Systems Frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[2] A. H. Ngu, M. Gutierrez, V. Metsis et al., "IoT middleware: a survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[3] O. Salman, I. Elhajj, A. Chehab, and A. Kayssi, "IoT survey: an SDN and fog computing perspective," *Computer Networks*, vol. 143, pp. 221–246, 2018.

[4] M. Stolpe, "The internet of things: opportunities and challenges for distributed data analysis," *ACM SIGKDD Explorations Newsletter*, vol. 18, no. 1, pp. 15–34, 2018.

[5] A. Sivieri, L. Mottola, and G. Cugola, "Building internet of things software with ELIoT," *Computer Communications*, vol. 90, pp. 141–153, 2016.

[6] S. Asano, T. Yashiro, and K. Sakamura, "Device collaboration framework in IoT-aggregator for realizing smart environment," in *Proceedings Of the 2016 IEEE TRON Symposium (TRONSHOW)*, pp. 1–9, Tokyo, Japan, December 2016.

[7] E. Blasch, A. Steinberg, S. Das et al., "Revisiting the JDL model for information exploitation," in *Proceedings of the 16th International Conference on Information Fusion*, pp. 129–136, Istanbul, Turkey, July 2013.

[8] X. Yong, Y. Fang, Y. Wu, and P. Yang, "An asynchronous sensor bias estimation algorithm utilizing targets' positions only," *Information Fusion*, vol. 27, pp. 54–63, 2016.

[9] D. E. George and A. Unnikrishnan, "On the divergence of information filter for multi sensors fusion," *Information Fusion*, vol. 27, pp. 76–84, 2016.

[10] J. A. Balazs and J. D. Velásquez, "Opinion mining and information fusion: a survey," *Information Fusion*, vol. 27, pp. 95–110, 2016.

[11] D. Brugali and A. Shakhimardanov, "Component-based robotic engineering (Part II)," *IEEE Robotics & Automation Magazine*, vol. 17, no. 1, pp. 100–112, 2010.

[12] A. Steck, A. Lotz, and C. Schlegel, "Model-driven engineering and run-time model-usage in service robotics," *ACM SIGPLAN Notices*, vol. 47, no. 3, pp. 73–82, 2011.

[13] F. A. A. Cheein and R. Carelly, "Agricultural robotics: unmanned robotic service units in agricultural tasks," *IEEE Industrial Electronics Magazine*, vol. 7, no. 3, pp. 48–58, 2013.

[14] A. Waugh, "Specifying metadata standards for metadata tool configuration," *Computer Networks and ISDN Systems*, vol. 30, no. 7, pp. 23–32, 1998.

[15] C. Fortuna, P. Oniga, Z. Padrah, M. Mohorcic, and A. Moraru, "Metadata management for the web of things: a practical perspective," *Proceedings Of the Third International Workshop On the Web Of Things (WoT '12)*, vol. 4, pp. 1–6, 2012.

[16] U. Hassan, M. Bassora, A. H. Vahid, S. O'Riain, and E. Curry, "A collaborative approach for metadata management for Internet of Things: linking micro tasks with physical objects," in *Proceedings Of the 9th International Conference On Collaborative Computing: Networking, Applications And Worksharing*, pp. 593–598, IEEE, Austin, TX, USA, October 2013.

[17] S. K. Datta and C. Bonnet, "Connect and control things: integrating lightweight IoT framework into a mobile application," in *Proceedings Of 9th International Conference On Next Generation Mobile Applications*, pp. 66–71, Services and Technologies, Cambridge, United Kingdom, September 2015.

[18] A. Hogan, M. Arenas, A. Mallea, and A. Polleres, "Everything you always wanted to know about blank nodes," *Journal of Web Semantics*, vol. 28, pp. 42–69, 2014.

[19] Web of things (WoT) thing description, 2020, https://w3.org/TR/wot-thing-description.

[20] E. Korkan, S. Kaebisch, M. Kovatsch, and S. Steinhorst, "Safe interoperability for web of things devices and systems," in *Languages, Design Methods, and Tools for Electronic System Design. Lecture Notes in Electrical Engineering*, T. J. Kazmierski, S. Steinhorst, and D. Große, Eds., pp. 47–69, Springer, Cham., Switzerland, 1st ed. edition, 2020.

[21] A. Haller, K. Janowicz, S. J. D. Cox et al., "The modular SSN ontology: a joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation," *Semantic Web*, vol. 10, no. 1, pp. 9–32, 2019.

[22] ISO/IEC 29100:2011, 2020, Available online: https://iso.org/standard/45123.html.

[23] K. Janowicz, A. Haller, S. J. D. Cox, D. le Phuoc, and M. Lefrançois, "SOSA: a lightweight ontology for sensors, observations, samples, and actuators," *Journal of Web Semantics*, vol. 56, pp. 1–10, 2019.

[24] K. Janowicz and M. Compton, "The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor Network ontology," in *Proceedings of the 3rd International Conference on Semantic Sensor Networks*, pp. 64–78, SSN '10, Shanghai, China, November 2010.

[25] H. Zorgati, R. B. Djemaa, I. A. B. Amor, and F. Sedes, "QoC enhanced semantic IoT model," in *Proceedings of the 24th Symposium on International Database Engineering & Applications*, pp. 1–7, Incheon, South Korea and Montreal, Canada, August 2020.

[26] A. Ciortea, S. Mayer, O. Boissier, and F. Gandon, "Exploiting interaction affordances: on engineering autonomous systems for the web of things," in *Proceedings Of the Second W3C Workshop On the Web Of Things*, Munich, Germany, June 2019.

[27] J. Ploennigs, A. Ba, and M. Barry, "Materializing the promises of cognitive IoT: how cognitive buildings are shaping the way," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2367–2374, 2018.

[28] S. D. Nagowah, H. B. Sta, and B. A. Gobin-Rahimbux, "Towards achieving semantic interoperability in an IoT-enabled smart campus," in *Proceedings Of the 2019 IEEE International Smart Cities Conference (ISC2)*, pp. 3–13, Casablanca, Morocco, October 2019.

[29] H. Rijgersberg, M. van Assem, and J. Top, "Ontology of units of measure and related concepts," *Semantic Web*, vol. 4, no. 1, pp. 3–13, 2013.

[30] A. Ciortea, S. Mayer, and F. Michahelles, "Repurposing manufacturing lines on the fly with multi-agent systems for the web of things," in *Proceedings Of the 17th International Conference On Autonomous Agents And MultiAgent Systems*, pp. 813–822, Stockholm, Sweden, July 2018.

[31] A. Umbrico, A. Cesta, G. Cortellessa, and A. Orlandini, "A holistic approach to behavior adaptation for socially assistive robots," *International Journal of Social Robotics*, vol. 12, no. 3, pp. 617–637, 2020.

[32] E. M. Sanfilippo, F. Belkadi, and A. Bernard, "Ontology-based knowledge representation for additive manufacturing," *Computers in Industry*, vol. 109, pp. 182–194, 2019.

[33] S. Mayer, N. Inhelder, R. Verborgh, R. van de Walle, and F. Mattern, "Configuration of smart environments made simple: combining visual modeling with semantic metadata and reasoning," in *Proceedings Of the International Conference On the Internet Of Things (IoT)*, Cambridge, MA, USA, October 2014.

[34] P. Rosenberger and D. Gerhard, "Context-awareness in industrial applications: definition, classification and use case," in *Proceedings of 51st CIRP Conference on Manufacturing Systems*, pp. 1172–1177, Stockholm, Sweden, May 2018.

[35] A. Pliatsios, C. Goumopoulos, and K. Kotis, "A review on IoT frameworks supporting multi-level interoperability – the semantic social Network of things framework," *International Journal on Advances in Internet Technology*, vol. 13, no. 1, pp. 46–64, 2020.

[36] F. Shi, Q. Li, T. Zhu, and H. Ning, "A survey of data semantization in internet of things," *Sensors*, vol. 1, no. 313, 2018.

[37] T. Elsaleh, S. Enshaeifar, R. Rezvani et al., "A lightweight ontology for internet of things data streams and its use with data analytics and event detection service," *Sensors*, vol. 20, no. 953, 2020.

[38] M. Bedworth and J. O'Brien, "The omnibus model: a new model of data fusion?" *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 4, pp. 30–36, 2000.

[39] A. N. Steinberg and C. L. Bowman, "Revisions to the JDL data fusion model," in *Proceedings of Sensor Fusion: Architectures, Algorithms, and Applications III (AeroSense '99)*, Orlando, FL, USA, April 1999.

[40] E. P. Blasch and S. Plano, "JDL level 5 fusion model "user refinement" issues and applications in group tracking," in *Proceedings of Signal Processing, Sensor Fusion, and Target Recognition XI (AeroSense 2002)*, Orlando, FL, USA, September 2002.

[41] P. C. Lin, H. Komsuoglu, and D. E. Koditschek, "Sensor data fusion for body state estimation in a hexapod robot with dynamical gaits," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 932–943, 2006.

[42] M. Cashmore, M. Fox, D. Long et al., "ROSPlan: planning in the robot operating system," in *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*, no. 1, Jerusalem, Israel, June 2015.

[43] H. Durmuş, E. O. Güneş, and M. Kırcı, "Data acquisition from greenhouses by using autonomous mobile robot," in *Proceedings of the 2016 Fifth International Conference on Agro-Geo Informatics*, pp. 1–5, Tainjin, China, July 2016.

[44] S. Zheng, Z. Lin, Q. Zheng, C. Liu, and H. Xiong, "IAPcloud: a cloud control platform for heterogeneous robots," *IEEE Access*, vol. 6, pp. 30577–30591, 2018.

[45] O. Vermesan, A. Bröring, E. Tragos et al., "Internet of robotic things – converging sensing/actuating, hyperconnectivity, artificial intelligence and IoT platforms," in *Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution*, O. Vermesan and J. Bacquet, Eds., pp. 97–155, River Publishers, Denmark, 1st ed. edition, 2017.

[46] C. Mahieu, F. Ongenae, F. de Backere, P. Bonte, F. De Turck, and P. Simoens, "Semantics-based platform for context-aware and personalized robot interaction in the internet of robotic things," *Journal of Systems and Software*, vol. 149, pp. 138–157, 2019.

[47] Robot operating system(ROS), 2020, https://www.ros.org.

[48] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: ROS for non-ROS users," in *Proceedings of the 15 International Symposium ISRR*, pp. 493–504, Flagstaff, AZ, USA, December 2011.

[49] MoveIt project in robot operating system (ROS), 2020, https://moveit.ros.org/robots.

[50] R. Limosani, A. Manzi, L. Fiorini, P. Dario, and F. Cavallo, "Connecting ROS and FIWARE: concepts and tutorial," in *Studies in Computational Intelligence Studies In Computational Intelligence and in Robot Operating System (ROS)*, D. Portugal, L. Iocchi, and A. Farinelli, Eds., Springer, Cham, Switzerland, pp. 449–475, 2019.

[51] L. Fiorini, G. D'Onofrio, E. Rovini et al., "A robot-mediated assessment of tinetti balance scale for sarcopenia evaluation in frail elderly," in *Proceedings of the 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–6, Delhi, India, October 2019.

[52] G. Karalekas, S. Vologiannidis, and J. Kalomiros, "Europa - a ROS-based open platform for educational robotics," in *Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, pp. 452–457, Metz, France, September 2019.

[53] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in *Proceedings of the International Symposium on Handheld and Ubiquitous Computing (HUC '99)*, pp. 304–307, Karlsruhe, Germany, September 1999.

[54] J. Han, Y. Cho, and J. Choi, "Context-aware workflow language based on web services for ubiquitous computing," *Computational Science and Its Applications - ICCSA 2005*, in *Proceedings of International Conference On Computational Science And its Applications (ICCSA 2005)*, pp. 1008–1017, Singapore, May 2005.

[55] Swagger: API Documentation & Design Tools for Teams, 2020, https://swagger.io/.