An Ontology Integrating the Open Standards of City Models and Internet of Things for Smart-City Applications

Chih-Yuan Huang, Yao-Hsin Chiang, and Fuan Tsai, *Member, IEEE*

***Abstract*—Smart city applications integrate the human, physical, and digital systems in a built environment with Internet of Things (IoT) resources, city models, and domain models. However, existing methods for the integration are suitable for individual applications and lack interoperability among application modules. This study analyzed existing integration strategies and developed an ontology for integrating the data modeling standards of the Open Geospatial Consortium (OGC) CityGML, IndoorGML, and SensorThings API. To cope with the broad definition of "things" in the IoT, the proposed ontology supports multiple views of things, including the a-building-as-a-thing, a-room-as-a-thing, an-opening-as-a-thing, and a-device-as-a-thing views. Thus, the proposed ontology relates information from these three standards and supports semantic queries. We demonstrated the proposed solution in smart home, smart security, smart health care, and fire evacuation systems. Overall, the proposed solution can facilitate the integration of standard-based IoT resources and city models to support smart city applications.**

***Index Terms*—City models, Internet of Things (IoT), ontology, open standard, smart city.**

I. Introduction

A. Background

CURRENTLY, over 50% of people worldwide live in urban areas. Increases in the urban population have placed considerable pressure on the infrastructure and environment of cities. Technological advances have led to increasing interest in smart cities in a variety of domains [1], [2]. Spatial information is a requirement for smart city services [3], and technologies for managing and processing spatial information are crucial components of smart city infrastructure [4]. Tasking and sensing capabilities supported by the Internet of Things (IoT) are crucial aspects of smart city infrastructure for capturing city dynamics and performing real-time actions [5].

Manuscript received 26 September 2021; revised 9 December 2021, 7 March 2022, and 15 April 2022; accepted 13 May 2022. Date of publication 30 May 2022; date of current version 7 October 2022. This work was supported in part by the Ministry of Interior, Taiwan, under Grant 110CCL031C, and in part by the Ministry of Science and Technology, Taiwan under Grant 107-2119-M-008-022 and Grant 108-2621-M-008-004-MY2. *(Corresponding author: Chih-Yuan Huang.)*

Chih-Yuan Huang and Fuan Tsai are with the Center for Space and Remote Sensing Research, National Central University, Taoyuan 320, Taiwan (e-mail: cyhuang@csrsr.ncu.edu.tw; ftsai@csrsr.ncu.edu.tw).

Yao-Hsin Chiang was with the Department of Civil Engineering, National Central University, Taoyuan 320, Taiwan. He is now with the Global Customer Service, MOXA Inc, Taipei 242032, Taiwan (e-mail: chsimon4@gmail.com).

Digital Object Identifier 10.1109/JIOT.2022.3178903

IoT resources and spatial information are generally formulated as independent data sets or services through the use of various protocol standards or data models. Open standards provide interoperable solutions for describing and sharing IoT resources and spatial information. However, insufficient research has examined the linkages between these standards. Thus, the current study focused on the integration of IoT resources and spatial information.

A geospatial data set usually describes the location (i.e., geometries with 2-D or 3-D coordinates), attribute (i.e., textual, numerical, categorical, and ordinal characteristics), and temporal (i.e., creation time, sampling time, or valid time) information of features [7]. Geospatial features in a smart city should follow clearly defined semantic classes to express their inherited attributes, constraints, and functions [8], [9]. For instance, city features, such as buildings, roads, rooms, trees, and bodies of water, should be interpreted and analyzed differently according to their semantic meanings. Therefore, in recent years, standards have been introduced for defining semantic-rich city characteristics—for example, the Open Geospatial Consortium (OGC) CityGML [10], OGC IndoorGML [11], LandXML [12], and building information modeling (BIM) [13]. From following these open standards, city features can be represented and utilized in an interoperable manner.

On the other hand, the IoT provides access to the dynamic information of a city through sensors and actuators. According to the International Telecommunication Union (ITU), the IoT is "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies" [14]. The IoT has been used in various fields, such as in smart homes [15], e-health [16], intelligent transport systems [17], and smart factories [18].

The main challenge in IoT development is the heterogeneity issue [19], [20]. Specifically, existing IoT systems are usually produced according to different communication protocols and proprietary data models by different manufacturers. Although information can be transmitted in each integrated system, information usually cannot be shared directly across different proprietary systems. This heterogeneity issue is also called the IoT silos, which seriously hinder the development of the IoT [21].

For example, an e-health service can access data from house environment sensors, human wearable devices, home automation appliances, and medical instruments to provide medical services on the basis of sensor observations. However, because these devices are produced by different manufacturers, the data from different devices follow different data models and are released through different protocols. Thus, data are essentially locked in individual closed systems, which are difficult to access and integrate with third-party applications [16]. To address the IoT heterogeneity problem, open standards can be followed for unifying IoT data models and communication protocols so that different IoT systems can easily exchange data and cooperate with each other.

To be specific, this study selected OGC SensorThings API [26] as well as CityGML [10] and IndoorGML [11] because they define semantic-rich and comprehensive data models for IoT resources and spatial information. However, applications usually require a customized design for integrating spatial information and IoT resources [6], which would result in redundant development and misunderstanding on the data sets. Therefore, this research aims on proposing a unified framework linking IoT resources and city models by their relationships that consequently supports various applications. To avoid possible misunderstanding, please note that the proposed idea can be applied to other suitable IoT and geospatial feature standards, such as the World Wide Web Consortium (W3C) Web of Things (WoT) [43], European Telecommunications Standards Institute (ETSI) smart applications reference (SAREF), ontology and extensions [44], and BIM.

B. Objective

In general, this research focused on the integration of IoT resources and city models for supporting smart city applications in an interoperable manner. Since open standards can enable the achievement of interoperability [25], IoT resources and city models should be integrated according to open standards.

This study adopted the semantic Web technology [27] to connect the classes and entities of city models and IoT resources [28] on the basis of an ontology [29]. The semantic Web can flexibly detail different relationships while retaining data source independence. Therefore, the present study developed an integration ontology to define the relationships among SensorThings API, IndoorGML, and CityGML for integrating IoT resources and city models. In particular, this study focused on the semantic integration of objects in indoor spaces, where geometric attributes (e.g., topological relationships) are not included. As displayed in Fig. 1, the proposed ontology integrates data from different resources to support various smart city applications. Furthermore, because "things" in the IoT have a flexible definition, ontologies are designed for various views of things to support different use cases.

The proposed ontology can be used to express the relationship between IoT resources and city models in resource description framework (RDF) format [30], enabling convenient querying through SPARQL Protocol and RDF Query Language (SPARQL) queries for supporting various applications. To examine the suitability of the proposed ontology, this research simulated different smart city applications.

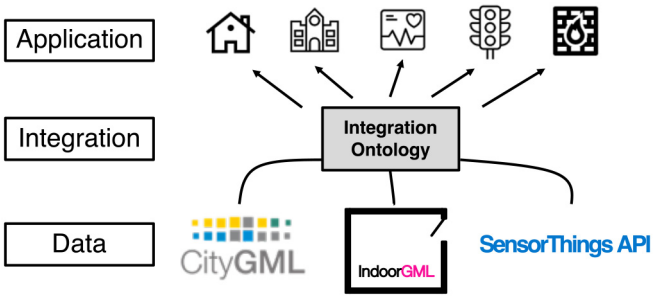


Fig. 1. Overall framework of the proposed solution.

In general, the contributions of this research are listed.

1) This research brings attention to the need of the linking standard-based IoT and city model and demonstrated it with different use cases.
2) Instead of creating proprietary classes for different use cases, this research follows only the classes from the chosen standards, which allows better interoperability.
3) As this research identified the relationships between IoT and city model classes based on their general nature instead of specific use cases or scenarios, the proposed solution has a broader applicability.
4) Although this research chose a certain integration strategy, the identification and analysis of different strategies could be helpful for people deciding the strategy for their systems/applications.

The following details the organization of the remaining sections of this article. Section II presents a review of studies on city models and IoT open standards and describes strategies for integrating these models and standards. Section III describes the proposed ontology for integrating SensorThings API, IndoorGML, and CityGML. Section IV provides the experimental results for the SPARQL queries used in the simulated smart city applications. Finally, Section V presents the research conclusions and suggestions for further investigations.

II. Related Work

Integrating IoT resources and city models is essential for smart city applications. This section presents a literature review on the integration of IoT resources and city models and introduces the open standards of city models and IoT resources. Furthermore, it describes strategies for integrating open standards.

A. Integrating 3-D Models and IoT Resources for Smart City

Cities should integrate cross-disciplinary components to become smart cities [31]. Numerous researchers have examined the integration of IoT sensors and 3-D models. For instance, Wang *et al.* [23] integrated indoor route network information and indoor sensor information to perform a dynamic risk assessment for planning immediate evacuation routes. Wang *et al.* [22] integrated an OGC sensor observation service (SOS) Web service and a BIM house model

TABLE I
COMPARISON OF DIFFERENT INTEGRATION STRATEGIES

STRATEGY	ADVANTAGES	DISADVANTAGES
Embedding	Atomic	1. Large data size 2. Inconvenience in updating dynamic information
External referencing	1. Lightweight 2. Referenced resource is independent 3. Can handle dynamic changes	Data are not self-contained
External joining	1. All resources are independent 2. Flexible many-to-many mappings 3. Can handle dynamic changes	May be unable to find suitable linkages between resources

in the application layer of a three-layer network architecture (which contained an application layer, a data service layer, and a data repository layer) for retrieving sensor observations from each room of the house. Chaturvedi *et al.* [32] integrated SOS services, CityGML, and historical solar energy observations to assess the solar radiation of building roofs and surfaces. In addition, the Dynamizer module proposed by [32] is being included in the CityGML 3.0 standard. Zhu *et al.* [24] integrated a spatiotemporal data series on air quality with a CityGML data set and an SOS service for visualizing air quality data with a city model.

However, the aforementioned studies did not consider interoperable standards and integrate resources in a customized manner for different applications. No study has provided a general solution for integrating IoT standards and city models.

B. Integration Strategies

Most relevant studies have integrated city models and IoT resources at the application level without following any open standards. Only a few studies have adopted open standard-based data sets. However, the integration methods of these studies are usually customized according to the target applications. Although data can be linked using the aforementioned methods, the methods are insufficiently general for adoption in other applications. Before designing a general solution for integrating city models and IoT resources, we analyzed and categorized the integration strategies adopted in previous studies.

From a literature review, three types of integration strategies were identified (Table I): the 1) embedding; 2) external referencing; and 3) external joining strategies. These strategies are described as follows.

1) In the embedding strategy, one resource is embedded directly into another resource; for example, a time series of sensor observations are embedded into city model data, such as the CityGML 3.0 Dynamizer [32]. In this case, all the data are atomic. Nevertheless, the data size is large and inconvenient for updating dynamic information.

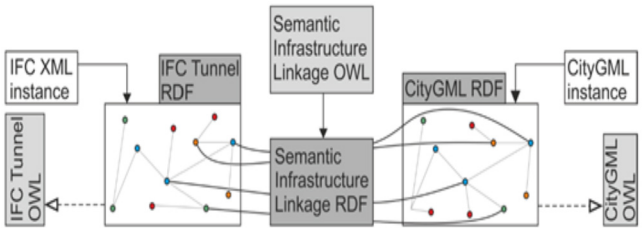


Fig. 2. Example of an external joining strategy involving the use of an RDF (with permission from ASCE) [34].

2) In the external referencing strategy, the relative or absolute path in one resource is used to retrieve another resource. For example, Kim *et al.* [33] derived the corresponding IndoorGML data set from a CityGML data set and then used a URI to back reference the IndoorGML data to the CityGML data. The external referencing strategy is more lightweight than the embedding strategy because every piece of information need not be embedded into one resource in the external referencing strategy. Moreover, dynamic information can be obtained through reference links in the external referencing strategy, such as the design in the OGC 3-D IoT platform for smart cities pilot [42]. This strategy also allows the referenced resources to be independent; thus, the referenced resources can be created and maintained as a single data set because they do not depend on any other resource. However, one minor drawback of the aforementioned strategy is that the data are not self-contained and may require connections to retrieve all the pieces of information.
3) In the external joining strategy, external data are created to describe and record the relationships between two resources. For example, Vilgertshofer *et al.* [34] defined a semantic linkage ontology and an RDF to integrate CityGML and industry foundation classes (IFC) data, as displayed in Fig. 2. To describe the relationships between resources, the URIs of specific IFC elements are mapped to the corresponding CityGML elements. This strategy allows all resources to be independent. The mapping relationships of these resources are relatively easy to update. The aforementioned strategy also enables the flexible processing of many-to-many mappings. However, one drawback of the aforementioned strategy is that it may be unable to find suitable linkages between resources because they are created independently.

External joining is sometimes achieved by semantic-based approaches, and some studies have performed cross-domain integration based on semantic frameworks. For example, Kuo and Hong [35] constructed an interoperable cross-domain semantic and geospatial framework for automatically detecting changed objects and regions. Peng and Goswami [36] proposed a methodology in which an ontology is used to integrate home environment data (e.g., humidity and temperature) and health data (e.g., the blood glucose level) from heterogeneous services and devices for health management applications.

TABLE II
COMPARISON OF RESOURCE INTEGRATION SOLUTIONS ON THE BASIS OF "EEEE" CRITERIA [37]

INTEGRATION METHODS	EFFECTIV ENESS	EXTENSIBI LITY	EFFORT	FLEXIBILI TY
New standards and models	case by case	case by case	case by case	case by case
Conversion, translation and extension of existing standards	medium	high	high	medium
Semantic web technologies	**high**	**high**	**high**	**medium**
Services-based methods	high	low	high	low
Application focused methods	case by case	low	low	Low

Liu *et al.* [37] reviewed existing resource integration solutions, and their findings are presented in Table II. The aforementioned authors found that semantic Web technologies are the most suitable technologies for resource integration. Thus, the semantic-based approach has the potential to meet the requirements for storing, sharing, and connecting heterogeneous data sets.

In summary, in terms of development, extensibility, and maintenance, we believe that the external joining strategy and semantic-based approach are the most suitable methods for achieving comprehensive and extensible integration of city models and IoT open standards. This research aimed to integrate cross-domain resources from CityGML, IndoorGML, and SensorThings API; accordingly, an integration ontology that maps these resources is proposed in this article. In the proposed ontology, users can effectively query all the information across the aforementioned open standard–based resources to establish extensible and complete cyber infrastructure for smart cities.

III. METHODOLOGY

This study integrated the open standards of OGC SensorThings API, IndoorGML, and CityGML and utilized the advantages of these standards to construct a smart city framework. This section describes the aforementioned open standards and the proposed resource integration method.

To represent data models precisely, namespaces are used to represent the resources of CityGML (i.e., citygml), IndoorGML (i.e., indoorgml), and SensorThings API (i.e., sta) classes and entities. Moreover, relationships (i.e., predicates) are represented in italic font.

A. OGC CityGML

OGC CityGML is a global open standard for city model information that represents the visual and geometric aspects of 3-D models of cities. The CityGML data model is sufficiently comprehensive and semantic rich for describing various thematic modules (e.g., bridges, tunnels, and buildings) and

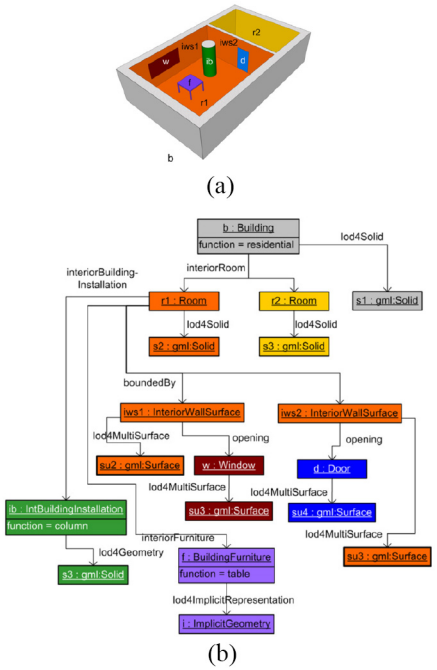


Fig. 3. Illustration of a CityGML LoD4 building: (a) spatial representation and (b) CityGML feature structure represented in the form of a UML diagram [38].

features (e.g., roofs, walls, windows, doors, and rooms) of a city. Moreover, five levels of detail (LoDs; from LoD0 to LoD4) are defined in the aforementioned model for various applications. Many city features, including the classes of building objects and relationships between each class, are defined in Fig. 3. As per the scope of this research, we focused on the `citygml:Building` class and its related classes. In particular, we examined the relationships between the classes of SensorThings API, IndoorGML, and CityGML LoD4.

The UML of CityGML includes the classes related to the citygml:Building class at LoD4 (e.g., `citygml:Room`, `citygml:IntBuildingInstallation`, `citygml:BoundarySarface`, `citygml:BuildingFurniture`, and `citygml:Opening`). The aforementioned classes are provided with geometric features to represent the geospatial information of objects, such as solid, multicurve, and multisurface. The UML diagram of CityGML also depicts the relationship between each class at LoD4 as well as the class's semantic information. For example, `citygml:Building` has *interiorRoom* `citygml:Room`, `citygml:BuildingFurniture`, has *interiorFuniture* of `citygml:Room`, `citygml:Room` is *boundedBy* `citygml:BoundarySurface`, `citygml:Room` has *roomInstallation* `citygml:IntBuilding Installation`, `citygml:BoundarySurface` has *opening* `citygml:Opening`. In addition, `citygml:Door` and `citygml:Window` inherit the class of `citygml:Opening`. According to the relationships proposed by CityGML, a data model with rich semantic relations was used in this research.

This research focused on the indoor space of buildings, including the objects inside the indoor space. Fig. 3 depicts

a CityGML LoD4 building based on the definition of CityGML. Each class of CityGML is described in the following text.

1) `citygml:Building`: This class contains buildings composed of structural segments.
2) `citygml:Room`: This class contains semantic objects that can be used to model the free space inside a building. Each semantic object should be associated with only one building part or building.
3) `citygml:BoundarySurface`: This class contains many thematic subclasses. The thematic features can be used to structure interior and exterior building installations, a building's exterior, and the visible surfaces of rooms.
4) `citygml:Opening`: This class contains objects that can be used for semantically describing openings, such as windows or doors, in interior or exterior boundary surfaces, such as roofs and walls.
5) `citygml:Window`: This class is used to model hatches between adjacent rooms or the windows in a building's exterior. The main difference between the `citygml:Door` and `citygml:Window` classes in normal cases is that the objects of the `citygml:Window` class are not specifically designed for the transit of vehicles or people.
6) `citygml:Door`: This class is used to model the doors located between rooms or in a building's exterior. The objects of this class can be utilized by people to leave or enter a room or building.
7) `citygml:BuildingFurniture`: The `citygml:Room` class may include objects from the `citygml:BuildingFurniture` and `citygml:IntBuildingInstallation` classes. The `citygml:BuildingFurniture` class contains movable objects in a room, including furniture.
8) `citygml:IntBuildingInstallation`: This class contains objects that are located inside a building and offer a special semantic meaning or function. In contrast to the objects of the `citygml:BuildingFurniture` class, the objects of the `citygml:IntBuildingInstallation` class are unmovable and inseparable from the building's structure. Objects in the `citygml:IntBuildingInstallation` class include railings, interior stairs, pipes, and radiators. The objects in this class are associated with objects from the `citygml:Building` or `citygml:Room` class.

B. OGC IndoorGML

CityGML focuses on the features of building components, including ceilings, roofs, walls, and floors, whereas IndoorGML [11] represents the indoor routing network between spaces (i.e., cells). In the IndoorGML data model, a cell is the basic space unit. Thus, IndoorGML offers a basic framework for representing the network, geometry, and semantics of cells within indoor spaces [39]. Indoor navigation applications, which are crucial smart city applications, can be realized with IndoorGML. Fig. 4 displays an example of an IndoorGML data set. In this figure, nodes and edges are used to form an indoor route network.

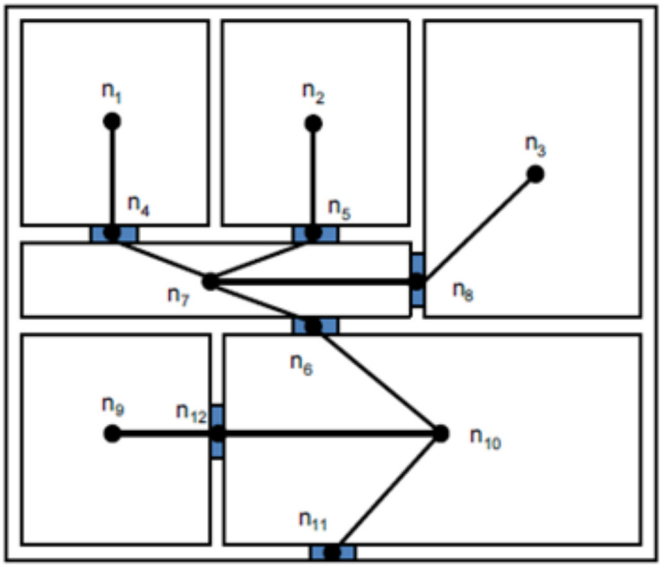


Fig. 4. Example of a topographic layer of IndoorGML [33].

The UML diagram of the IndoorGML core module illustrates the relationships between individual classes as well as their semantic information. We used the navigation functionality based on an IndoorGML route network, which includes the `indoorgml:State` and `indoorgml:Transition` classes.

The `indoorgml:State` class represents nodes, such as the doors, corridors, and rooms within a building. The objects in the `indoorgml:State` class are represented geometrically as points in IndoorGML. The entities of the `indoorgml:Transition` class are edges that represent the connectivity or adjacency relationship between two `indoorgml:State` entities, such as doors, stairs, and elevators. The weight attribute of `indoorgml:Transition` indicates the status of connectivity. The entities of the `indoorgml:Transition` class are represented as curved primitive objects in the OGC Geography Markup Language.

The IndoorGML data model also includes the semantic information between classes; that is, an `indoorgml:Transition` entity connects two `indoorgml:State` entities. Because this research focused on the indoor space of buildings, rooms were regarded as `indoorgml:State` entities and doors and stairs were regarded as `indoorgml:Transition` entities.

C. OGC SensorThings API

OGC SensorThings API is an IoT open-standard Web service. A comprehensive model for IoT resources that contains numerous classes and attributes, including tasking and sensing capabilities, is defined by SensorThings API. Currently, the SensorThings API standard comprises two parts: part 1, which is related to sensing capabilities (Fig. 5), and part 2, which is related to tasking capabilities (Fig. 6). Each class in the SensorThings API standard is described in the following text.

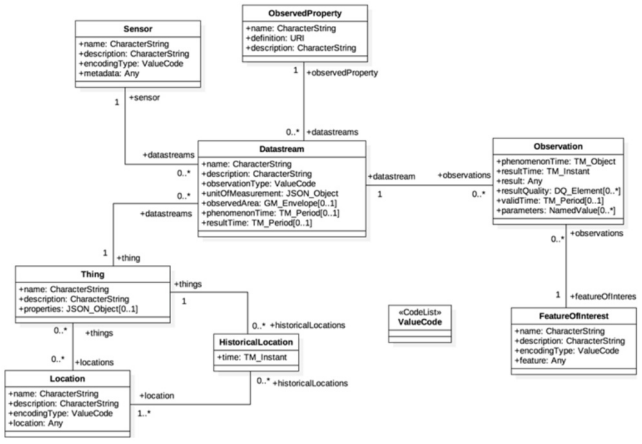


Fig. 5. Data model of OGC SensorThings API part 1 [26].

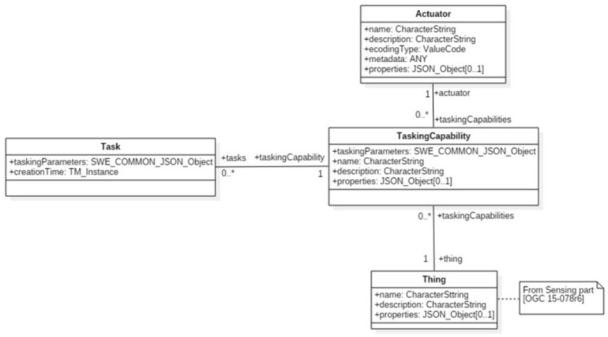


Fig. 6. Data model of OGC SensorThings API part 2 [40].

1) `sta:Thing`: The entities of the sta:Thing class are objects in the information domain (virtual things) or physical domain (physical things) that can be identified and integrated into communication networks [14].
2) `sta:Location`: The entities of this class record the last known location of the entities of the `sta:Thing` class.
3) `sta:HistoricalLocation`: The entities of this class record the time period or time points of previous locations of the entities of the `sta:Thing` class. For example, if a "thing" is mobile, several entities of the `sta:Location` class are linked to entities of the `sta:HistoricalLocation` class.
4) `sta:Datastream`: This class comprises entities of the `sta:Observation` class that measure the same entity of the `sta:ObservedProperty` class and are produced by the same entity of the `sta:Sensor` class. For instance, if an entity of the `sta:Thing` class is capable of observing three properties, such as illumination, relative humidity, and air temperature, then this entity may correspond to three `sta:Datastream` entities, each of which groups the `sta:Observation` entities for one feature.
5) `sta:Sensor`: The entities of this class represent the instruments used to monitor a phenomenon or property.
6) `sta:ObservedProperty`: The entities of this class represent the monitored properties, including illumination, relative humidity, and air temperature.


```
{
  "@iot.id": 54772365,
  "phenomenonTime": "2022-04-14T15:50:16.000Z",
  "result": 37,
  "resultTime": "2022-04-14T15:50:16.000Z",
  "@iot.selfLink": "https://sta.ci.taiwan.gov.tw/STA_AirQuality_v2/v1.0/Observations(54772365)",
  "Datastream": {
    "description": "細懸浮微粒 PM2.5",
    "@iot.id": 6049,
    "name": "PM2.5",
    "observationType": "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
    "observedArea": {
      "type": "Point",
      "coordinates": [
        120.459,
        23.121
      ]
    },
    "phenomenonTime": "2022-04-14T04:02:20.000Z/2022-04-15T13:35:48.000Z",
    "resultTime": "2022-04-14T04:02:20.000Z/2022-04-15T13:35:48.000Z",
    "@iot.selfLink": "https://sta.ci.taiwan.gov.tw/STA_AirQuality_v2/v1.0/Datastreams(6049)",
    "unitOfMeasurement": {
      "name": "microgram per cubic meter",
      "symbol": "μg/m3",
      "definition": "https://acronyms.thefreedictionary.com/ug%2Fm3"
    }
  },
  "FeatureOfInterest@iot.navigationLink": "https://sta.ci.taiwan.gov.tw/STA_AirQuality_v2/v1.0/
Observations(54772365)/FeatureOfInterest",
  "MultiDatastream@iot.navigationLink": "https://sta.ci.taiwan.gov.tw/STA_AirQuality_v2/v1.0/
Observations(54772365)/MultiDatastream",
  "Datastream@iot.navigationLink": "https://sta.ci.taiwan.gov.tw/STA_AirQuality_v2/v1.0/
Observations(54772365)/Datastream"
}
```


Fig. 7. Example query of SensorThings API.

7) `sta:Observation`: The entities of this class represent the determined or measured value of a property represented by an entity of the `sta:ObservedProperty` class that is measured by an entity of the `sta:Sensor` class.
8) `sta:FeatureOfInterest`: This class comprises the features corresponding to the entities of the `sta:Observation` class.
9) `sta:TaskingCapability`: This class comprises the controllable capabilities supported by entities of the `sta:Thing` class.
10) `sta:Task`: This class contains user commands for controlling entities of the `sta:TaskingCapability` class. Device control should be performed on the basis of the input values contained in the `sta:Task` class.
11) `sta:Actuator`: This class contains metadata of the instrument used for obtaining the entities of the `sta:TaskingCapability` class.

SensorThings API hosts IoT resources in the RESTful Web service style and JSON format. An example of a query result is displayed in Fig. 7. An entity of the `sta:Thing` class contains numerous attributes, including properties, a description, and a name. SensorThings API contains the aforementioned attributes as well as navigation links that connect related entities.

Generally speaking, SensorThings API is a comprehensive solution for an IoT Web service. A general and complete data model is defined in the aforementioned standard for IoT tasking and sensing. Moreover, to enable users to query targeted IoT resources, SensorThings API uses flexible query functions and the RESTful Web service style.

On the basis of the semantic information obtained from the SensorThings API data model, we constructed a lightweight ontology of SensorThings API, called the STA-LITE[1] ontology. The STA-LITE ontology describes semantic classes and the relationships between them. For example, the `sta:Thing`

[1]STA-LITE ontology: https://gitlab.com/gaialab.ncu/iot-and-city-model-standard-integration/-/blob/main/STA-Lite.owl.

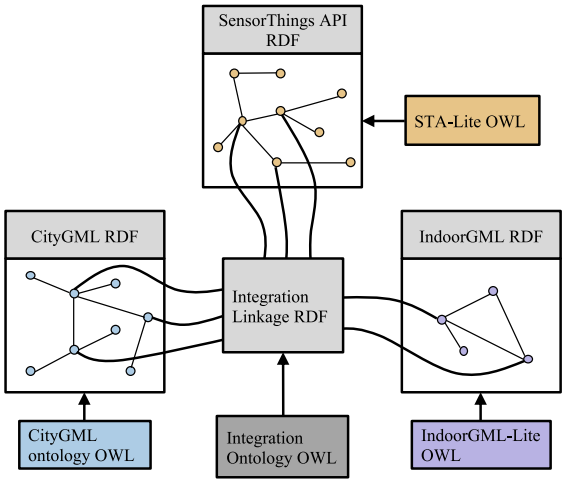


Fig. 8. Integration framework proposed in this article.

and `sta:Datastream` classes have a *hasDatastream* relationship. Moreover, the `sta:Thing` and `sta:TaskingCapability` classes have a *hasTaskingCapability* relationship.

D. Integration Strategy

This study adopted the semantic Web technology for integrating the SensorThings API, IndoorGML, and CityGML data models. The relationships between data from various domains can be flexibly and effectively described with the semantic Web technology.

"Things" can be defined differently according to the application, which creates a unique challenge in integrating city models and IoT resources. For example, the sta:Thing class can contain entities, such as doors, rooms, corridors, windows, appliances, and sensors. Therefore, different definitions of "things" should be considered during the integration of IoT resources and city models.

The semantic Web can express the relationships between resources and can integrate resources by linking their URIs; thus, independent data can be generated. An integration ontology capable of defining the relationships among the SensorThings API, IndoorGML, and CityGML data models is crucial for integrating IoT resources and city models. By developing such an integration ontology, data providers and users can determine the relationships between IoT resources and city models as well as perform cross-domain queries. In this research, three semantic Web standards were followed: ontology Web language (OWL), the RDF, and SPARQL.

The integration framework proposed in this study is shown in Fig. 8. A comprehensive data model with high extensibility and rich semantic information was used to individually record data from different resources in the RDF format according to their individual ontologies, such as the CityGML, IndoorGML-Lite, and STA-Lite ontologies. The following section details these ontologies. All the attributes and information of each piece of data (e.g., name, geometry, and observations) were completely recorded in an RDF file. The RDF describes relationships in the form of triples (i.e., subject–predicate–object). According to the triples structure, the current study developed an integration ontology that characterizes the relationships between resources from the aforementioned three ontologies (Fig. 8).

1) Properties of the Resources in the Integration Ontology Framework: In the proposed integration ontology, the relationship between sta:Observation and sta:TaskingCapability (i.e., *observes* and *hosts*, respectively) is described according to the Semantic Sensor Network Ontology[2] published by the W3C in 2017. W3C published another ontology called building topology ontology (bot) in 2019. The predicate of the bot (i.e., *containsElement*) is applied to the relationship that a zone or space (such as a room or building) contains an entity or element (e.g., a device or node).

The CityGML ontology[3] was developed by Métral *et al.* [41]. We also constructed the indoorgml-lite[4] ontology to describe the relationships between the `indoorgml:State` and `indoorgml:Transition` classes of IndoorGML (i.e., *connects*) as well as the `indoorgml:State` and `indoorgml:MultiLayerGarph` classes of IndoorGML (i.e., *interConnects*). The presented STA-LITE ontology describes the essential relationships between the classes of SensorThings API, such as *hasTaskingCapability*, *hasDatastream*, *hasObservation*, and *hasObservedProperty*.

In addition, we constructed suitable ontologies for defining relationships that are not described in existing ontologies, which is represented with the prefix *cgis*.[5] The defined relationships include *isThing*, *isState*, *isTransition*, *isMultiLayerGraph*, *happensIn*, *withinCellspaceOf*, and *enables*. These predicates are introduced in the following sections.

2) Multiple Definitions of Things in IoT: The relationships between the classes of the SensorThings API, IndoorGML, and CityGML data models can be expressed with the proposed integration ontology. Depending on the application, different views of sta:Thing entities should be considered, including the a-device-as-a-thing, an-opening-as-a-thing, a-room-as-a-thing, and a-building-as-a-thing views. In Figs. 9–12, CityGML, IndoorGML, and SensorThings API classes are displayed in blue, lavender, and yellow, respectively.

a) A-device-as-a-thing view: Fig. 9 illustrates the a-device-as-a-thing view, in which the `sta:Thing` class in SensorThings API is mapped to the `citygml:BuildingFurniture` class. A CityGML resource can be directly linked with the `sta:Thing` class through the *isThing* predicate if the resource directly corresponds to an entity of the `sta:Thing` class. Additionally, the proposed integration ontology supports the relationships between a `sta:Thing` entity and a CityGML feature (which *hosts* the `sta:Thing` feature), such as `citygml:BoundarySarface`, `citygml:Room`,

[2]Semantic Sensor Network Ontology:https://www.w3.org/TR/vocab-ssn/# SOSA3 Building Topology Ontology: https://w3c-lbd-cg.github.io/bot/.

[3]CityGML ontology: http://cui.unige.ch/isi/onto//citygml2.0.owl.

[4]indoorgml-lite ontology: https://gitlab.com/gaialab.ncu/iot-and-city-model-standard-integration/-/blob/main/IndoorGML-Lite.owl.

[5]cgis ontology: https://gitlab.com/gaialab.ncu/iot-and-city-model-standard-integration/-/blob/main/cgis.owl.

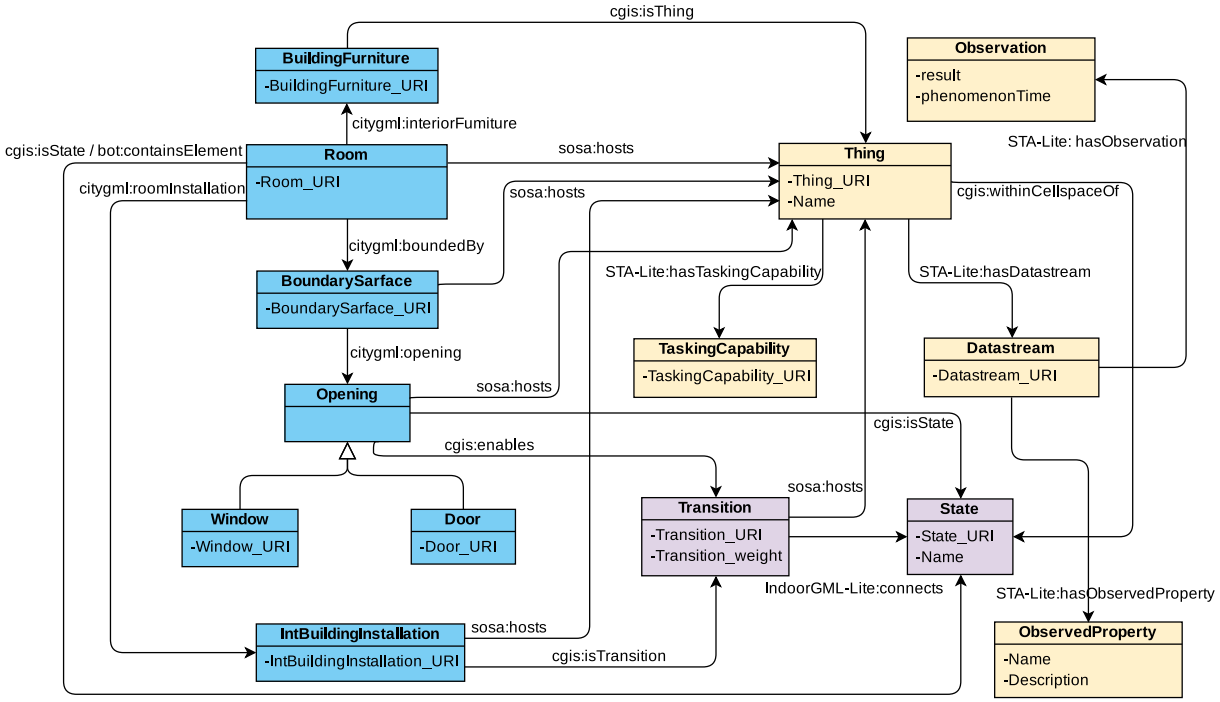


Fig. 9. Integration ontology for the a-device-as-a-thing view.

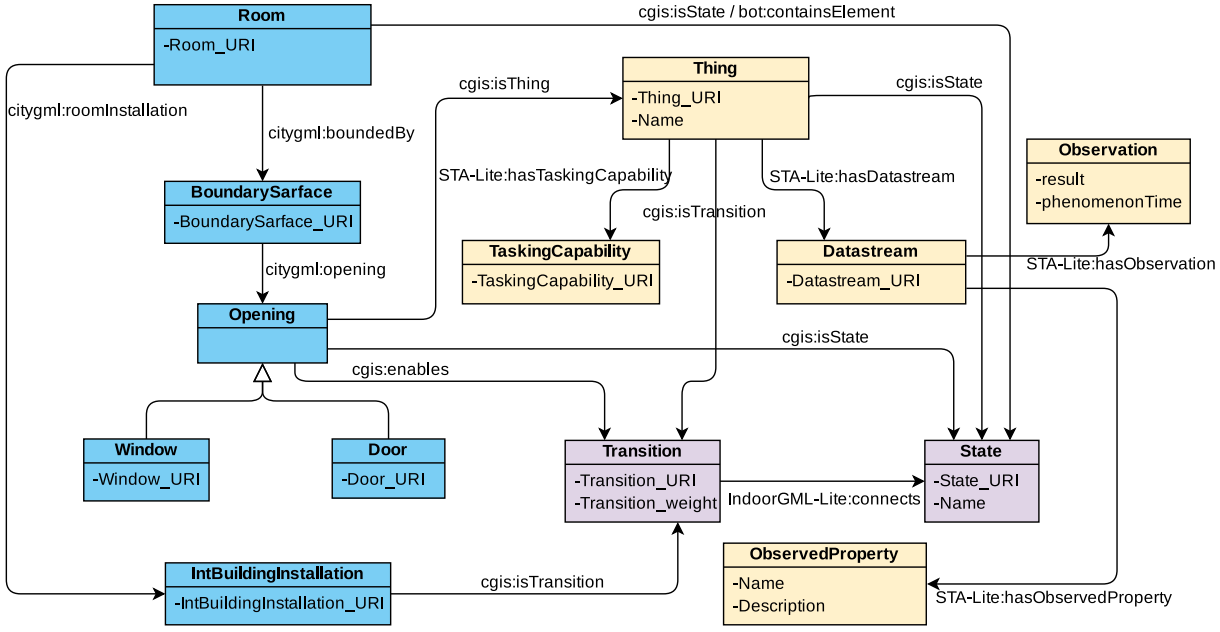


Fig. 10. Integration ontology of the an-opening-as-a-thing view.

`citygml:IntBuildingInstallation`, `indoorgml:Transition`, and `citygml:Opening`, even if there is no `sta:BuildingFurniture` feature directly mapped to the `sta:Thing` entity.

With regard to the relationship between the `citygml:Room` and `indoorgml:State` classes, an entity of the `citygml:Room` class can be represented as one node (i.e., *isState*) or several nodes (i.e., *containsElement*). The current research considers `indoorgml:State` as the entity of a cell; thus, when an entity of the `sta:Thing` class is located within the space of an entity of the `indoorgml:State` class, `sta:Thing` is *withinCellspaceOf* `indoorgml:State`. Moreover, `citygml:Opening` *enables* `indoorgml:Transition` (i.e., a door or a window).

b) An-opening-as-a-thing view: Fig. 10 depicts the an-opening-as-a-thing view, in which an entity of the `citygml:Opening` class (i.e., an entity of the `citygml:Window` or `citygml:Door` class) is an entity of the `sta:Thing` class. In the aforementioned view, `sta:Thing` can be recognized as a node (i.e., `indoorgml:State`) or an edge (i.e., `indoorgml:Transition`) according to the class

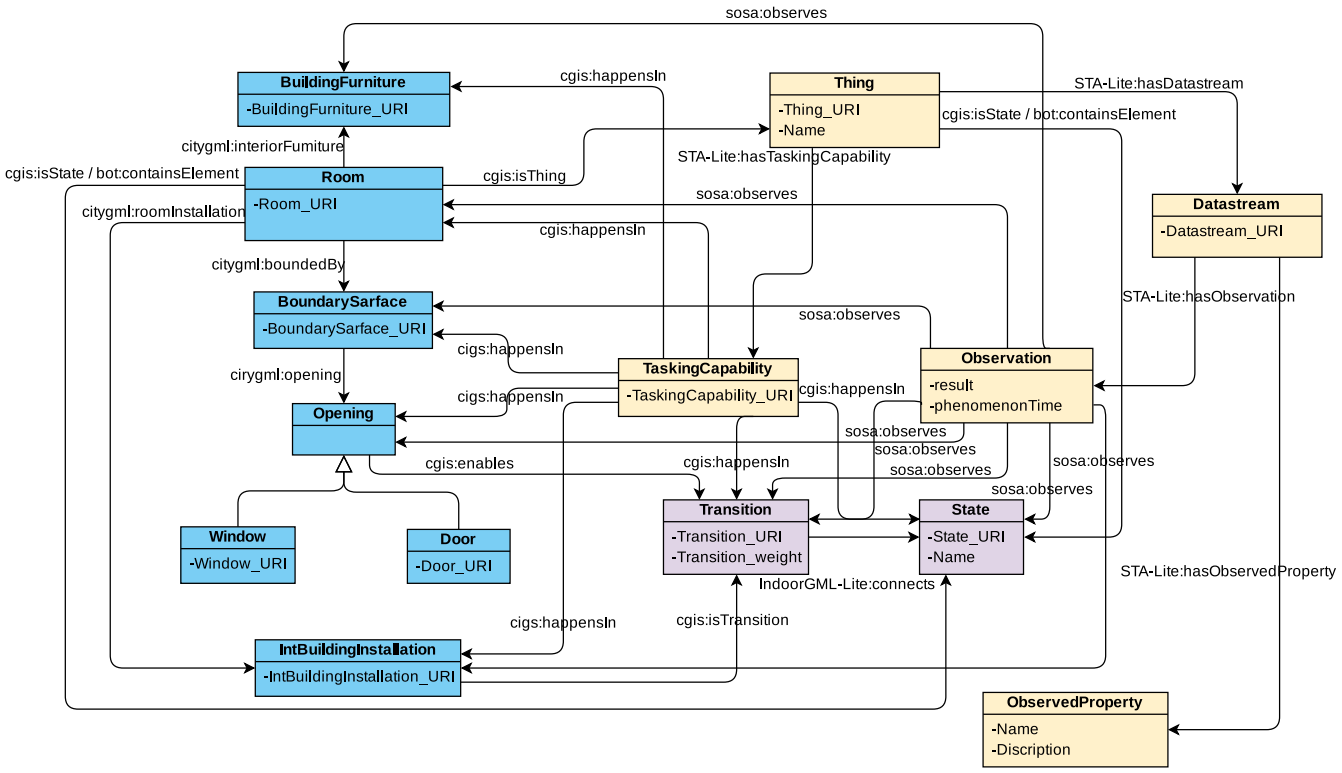


Fig. 11. Integration ontology of the a-room-as-a-thing view.

definitions in IndoorGML. In the an-opening-as-a-thing view, an entity of the `sta:Datastream` class, which corresponds to an entity from the sta:Thing class, is capable of recording the state (e.g., closed or open) or additional information (e.g., the passage size) of the `citygml:Opening` class. The entities of the `sta:TaskingCapability` class represent the controllable actions that can be performed by a window or door actuator. The proposed ontology can also describe the relationships between the entities of the `sta:Thing` class and other CityGML entities.

c) A-room-as-a-thing view: Fig. 11 depicts the a-room-as-a-thing view. In the aforementioned view, an entity of the `sta:Observation` class *observes* a phenomenon that occurs in a room. The room is represented as a `sta:Thing` entity within the zone space that contains some `indoorgml:State` entities. The `sta:Datastream` entities comprise a set of `sta:Observation` entities for a certain `sta:ObservedProperty` entity.

Regarding the IndoorGML classes, a `sta:Tasking Capabiltiy` might also *happensIn* a `indoorgml:State` or `indoorgml:Transition` (e.g., locking or unlocking a door). For the relationships between the CityGML and IndoorGML classes, a `citygml:Room` entity may contain some `indoorgml:State` entities that cover a small geometrical space.

In particular, a `sta:Thing` entity may contain various entities (e.g., a thermometer, a buzzer, a door, a window, or the room) with sensing or tasking capabilities in the room. Thus, `sta:TaskingCapabilities` (e.g., turning an alarm on or off or changing the illumination) also *happensIn* the entities, and `sta:Observations` (e.g., relative humidity and air temperature) *observes* those entities.

d) A-building-as-a thing view: Fig. 12 illustrates the a-building-as-a-thing view, in which buildings comprise numerous components (e.g., `citygml:Opening`, `citygml:BuildingFurniture`, and `citygml:Room`). The concept of a building containing numerous components and rooms has similarity with the a-room-as-a-thing view, in which several entities constitute the `sta:Thing` class. Therefore, in the a-building-as-a-thing view, a `sta:Observation` of a `sta:Thing` *observes* a phenomenon that could occur in a room, the building, or any entity of the building. Moreover, a `sta:TaskingCapability` *happensIn* a specific entity of the building (e.g., a room, a door, or an air conditioner). The relationships between the `sta:TaskingCapability` and sta:Datastream classes in the a-building-as-a-thing view are similar to those in the a-room-as-a-thing view.

Furthermore, entities of the `indoorgml:MultiLayer Garph` class, which represent different types of spaces in a building, can correspond to different building features. Entities from the `indoorgml:MultiLayerGarph` class can form a group of space layers in different domains. For simplicity, this research theorizes that an entity of the `citygml:Building` class can represent an entity of the `indoorgml:MultiLayerGarph` class (i.e., *isMultiLayerGraph*).

The proposed integration ontology is saved in RDF format. When this ontology is applied, IoT resources and city models can be integrated and presented in RDF format. Information obtained from different providers can be stored independently in a uniform manner on the basis of open standards. Furthermore, the semantic Web technology's flexibility allows

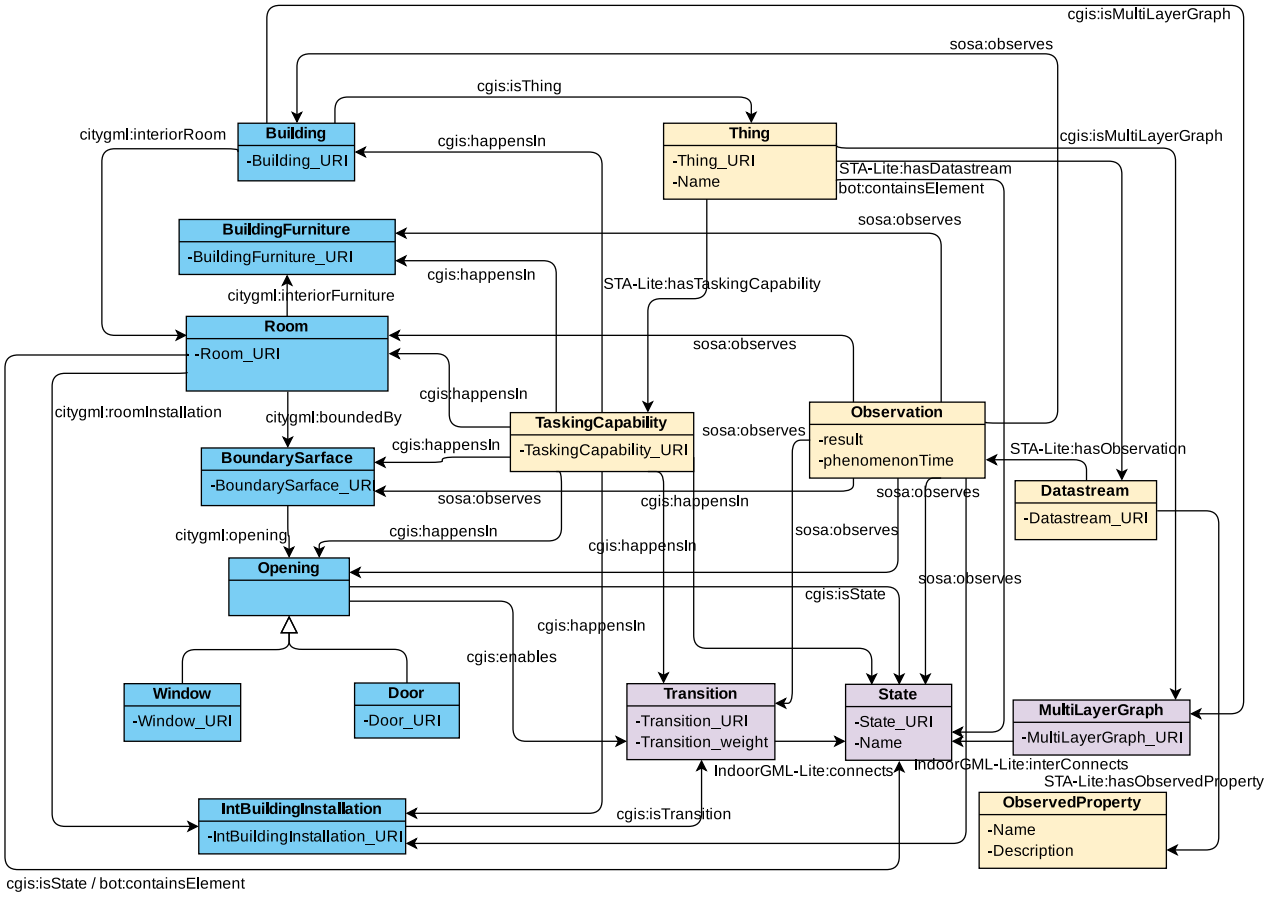


Fig. 12. Integration ontology of the a-building-as-a-thing view.

the incorporation of the relationships in multiple `sta:Thing` views into the same data set, thus providing support for additional smart city applications.

IV. Implementation Results

To verify the suitability of the proposed ontology, we adopted use cases requiring queries that link IoT resources and city models. In the proposed integration ontology, users can adopt SPARQL queries to obtain data. The use cases adopted for a smart health care system and a fire evacuation system are described in the following sections to prove the concept.

A. Testing Data Set of 3-D City Model

The data set of the R3 building in National Central University, Taiwan was adopted as the testing data set in this study. This research constructed the entities of the building and the indoor route network for the city model (Figs. 13 and 14), including the features defined in the CityGML, IndoorGML, and SensorThings API data models and integrated RDF data.[6]

[6]R3 CityGML RDF: https://gitlab.com/gaialab.ncu/iot-and-city-model-standard-integration/-/blob/main/R3_Building.rdf R3 IndoorGML RDF: https://gitlab.com/gaialab.ncu/iot-and-city-model-standard-integration/-/blob/main/R3_Route.rdf SensorThing API RDF: https://gitlab.com/gaialab.ncu/iot-and-city-model-standard-integration/-/blob/main/R3_STA.rdf Integration RDF:https://gitlab.com/gaialab.ncu/iot-and-city-model-standard-integration/-/blob/main/R3_Integration.rdf.

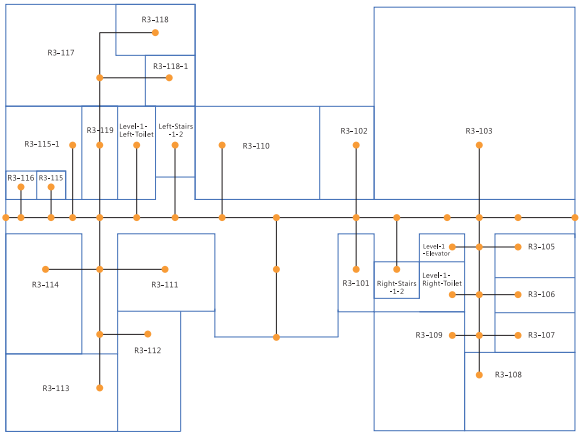


Fig. 13. Floor plan of the first floor of the study area.

B. Use Case Implementation

To illustrate the importance of integrating city models and IoT resources, we implemented several simulated use cases in which the role of CityGML was to provide geospatial information (i.e., the geometry of objects, such as a room, door, or window). In addition, IndoorGML forms indoor route networks that support navigation applications. Furthermore, the tasking and sensing capabilities of a SensorThings API service are used for remotely controlling and monitoring features, respectively.

Query 1 SPARQL Query for Retrieving the Physiological Information of Patients

```
PREFIX sta-lite: <http://140.115.110.71/sta-lite.ow1#>

select ?name ?HeartRate_result ?HeartRate_time ?SpO2_result ?SpO2_time
Where {
   ?Patient sta-lite:hasDatastream ?Datastream;
            sta-lite:name ?name
{
      ?Datastream sta-lite:hasObservedProperty/sta-lite:description "Heart
                                rate";
                  sta-lite:hasObservation ?HeartRate_ob.
      ?HeartRate_ob sta-lite:result ?HeartRate_result;
                  sta-lite:phenomenonTime ?HeartRate_time
}UNION{
   ?Datastream sta-lite:hasObservedProperty/sta-lite:description "SpO2";
               sta-lite:hasObservation ?SpO2_ob.
   ?SpO2_ob sta-lite:result ?SpO2_result;
            sta-lite:phenomenonTime ?SpO2_time
}
}
order by (?SpO2_time || ?HeartRate_time)
```

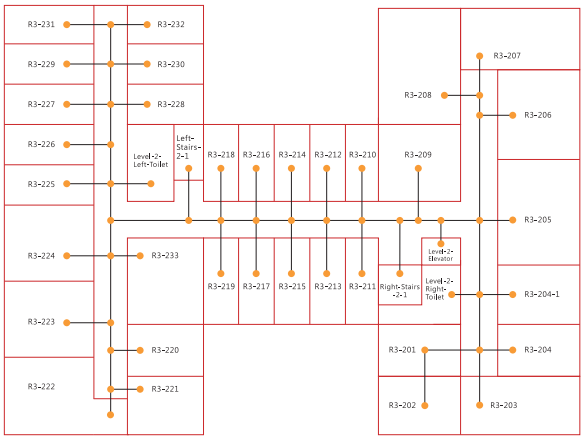


Fig. 14. Floor plan of the second floor of the study area.

TABLE III
RESULT FROM QUERY 1

name	HeartRate_result	HeartRate_time	SpO2_result	SpO2_time
Simon	91.2	2020-05-11T 00:11:00+08:00		
Simon			72.0	2020-05-11T 00:11:00+08:00

In the simulations, data integration was realized through the adoption of external RDF data obtained using the proposed integration ontology. The integration results of this study were verified with SPARQL queries.

1) Smart Health Care: In the smart health care use case, a patient' wearable device is considered a `sta:Thing` entity, which could also be seen as the patient. The wearable device on the patient produce three sta:Datastream entities from a heart rate sensor, an oximeter, and a radio frequency identification (RFID) tag. The heart rate sensor and oximeter record and monitor the physiological conditions of the patient. The RFID tag helps in recording the location of the patient (e.g., room name).

Query 1 retrieves the latest observation results of the heart rate sensor and oximeter (Table III). The observations indicate that a smart health care system can assist medical care personnel in monitoring the physiological condition of patients. Moreover, if the condition of a patient is abnormal, a smart health care system triggers a query to retrieve the location of the patient and activates a buzzer (i.e., an alarm) near the patient.

Query 2 SPARQL Query for Retrieving the Patient Location

```
PREFIX sta-lite: <http://140.115.110.71/sta-lite.ow1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX citygm1: <http://cui.unige.ch/isi/onto//citygm12.0.ow1#>
PREFIX cgis: <http://140.115.110.71/cgis.ow1#>

select ?RFID_result ?RFID_time ?State_of_Patient ?TaskingCapability
where{
  ?Patient sta-lite:name : "Simon"
           sta-lite:hasDatastream ?Datastream_RFID.
   ?Datastream_RFID sta-lite:hasObservedProperty/sta-lite:description "RFID
Location";
                    sta-lite:hasObservation ?RFID_ob.
?RFID_ob sta-lite:result ?RFID_result;
             sta-lite:phenomenonTime ?RFID_time.
?Room rdf:type citygml:Room.
        FILTER regex(str(?Room), ?RFID_result)
?Room cgis:isState ?State_of_Patient;
          cgis:isThing/sta-lite:hasTaskingCapability ?TaskingCapability;
          cgis:isThing/sta-lite:hasTaskingCapability/sta-lite:description
 "Buzzer control"
}
order by ?RFID_time
limit 1
```

TABLE IV
RESULT FROM QUERY 2

RFID_ result	RFID_time	State_of_Patient	TaskingCapability
R3-205	2020-05-11T00:11:00+08:00	http://140.115.110.71/R3_Route.owl#Node-92	http://140.115.110.71/R3_STA.owl#R3-205_Buzzer

Query 3 SPARQL Query for Retrieving AED Positions

```
PREFIX cgis: <http://140.115.110.71/cgis.ow1#>

select ?AED ?node
where {
      ?AED cgis:withinCellspaceOf ?node
      FILTER regex(str(?AED), "AED")
}
```

Query 2 searches for the latest RFID location of the patient (Table IV). Lines 14–18 are related to connecting the RFID location to a citygml:Room entity and an indoorgml:State entity as well as searching for a buzzer in the citygml:Room entity. The example result in Table IV indicates that the location of the patient "Simon" was near citygml:Room "R3-205" and indoorgml:State "Node-92" at 00:11 on May 11, 2020. The aforementioned table also indicates the `sta:Tasking Capability` entity of a buzzer that can help trigger an alarm. Finally, the indoorgml:State can help navigate medical care personnel to the patient while continuously monitoring physiological information.

In addition, for emergency situations, the smart health care system can help retrieve the positions of automated external defibrillators (AEDs). From a search for the `indoorgml: State` entities of AEDs (Query 3 and Table V), the shortest path from the patient (i.e., Node-92) to an AED can be calculated with the route network.

Query 4 SPARQL Query for Retrieving the Real-Time Observation Results of Smoke and Temperature Sensors

```
1  PREFIX sta-lite: <http://140.115.110.71/sta-lite.ow1#>
2
3  select ?Thing_smoke ?result_smoke ?time_smoke
          ?Thing_temp ?result_temp ?time_ temp
4  where {
5    {
6     ?Thing_smoke sta-lite:hasDatastream ?Datastream_smoke.
7     ?Datastream_smoke sta-lite:hasObservedProperty/sta-lite:description
    "Smoke Detector";
8              sta-lite:hasObservation/sta-lite:phenomenonTime ?time_smoke;
9              sta-lite:hasObservation/sta-lite:result ?result_smoke.
10             FILTER (?result_smoke > 2000)
11   }UNION{
12    ?Thing_temp sta-lite:hasDatastream ?Datastream_temp.
13    ?Datastream_temp sta-lite:hasObservedProperty/sta-lite:description
    "Temperature observation";
14             sta-lite:hasObservation/sta-lite:phenomenonTime ?time_temp;
15             sta-lite:hasObservation/sta-lite:result ?result_temp.
16             FILTER (?result_temp > 50)
17   }
18 }
19 order by (?time_temp || ?time_smoke)
```

TABLE V
RESULT FROM QUERY 3

AED	node
http://140.115.110.71/R3_Integration.owl#AED-1	http://140.115.110.71/R3_Integration.owl#Node-46
http://140.115.110.71/R3_Integration.owl#AED-2	http://140.115.110.71/R3_Integration.owl#Node-52
http://140.115.110.71/R3_Integration.owl#AED-3	http://140.115.110.71/R3_Integration.owl#Node-75
http://140.115.110.71/R3_Integration.owl#AED-4	http://140.115.110.71/R3_Integration.owl#Node-106

2) Fire Evacuation System: In the fire evacuation system use case, IoT devices can provide real-time smoke density and temperature observations and city models can provide spatial information and indoor navigation routes. To construct an effective fire evacuation system, city models and IoT resources must be integrated.

Query 4 retrieves real-time observations of smoke density and temperature and checks if the results are abnormal. Lines 10 and 19 set thresholds for the observation results. If the smoke density is higher than 2000 ppm or the temperature is higher than 50 °C, then the fire evacuation system considers the situation to be an emergency. SPARQL Query 4 returns the data to the system if any result meets the thresholds. Table VI displays a situation in which the smoke density and temperature results meet the thresholds. In this case, the system triggers an SPARQL query to retrieve the corresponding `sta:TaskingCapability` entities of sprinklers and ventilation systems.

Lines 8 and 9 in Query 5 find the room associated with the `sta:Thing` entity that *hosts* the temperature sensor that obtains high readings. Lines 10–13 search for the indoorgml:State entity that represents the aforementioned room and other indoorgml:State entities that *connect* with this room to expand the query area for finding alarms. Lines 14–16 search for the `sta:TaskingCapability` entity of any sprinkler associated with an indoorgml:State entity of the aforementioned room. Lines 17–19 search for the `sta:TaskingCapability` of any buzzer associated with the surrounding indoorgml:State entities.

SPARQL Query 6 also uses the aforementioned search procedure to find ventilation systems associated with the `indoorgml:State` entities of the room with high smoke density readings. According to the proposed integration solution, the `sta:TaskingCapability` entities of sprinklers,

TABLE VI
RESULT FROM QUERY 4

Thing_smoke	result_smoke	time_smoke
http://140.115.110.71/R3_STA.owl#Device_SmokeSensor	3214.0	2020-05-28T20:47:12+08:00
Thing_temp	**result_temp**	**time_temp**
http://140.115.110.71/R3_STA.owl#Device_TemperatureSensor	72.1	2020-05-28T20:41:00+08:00

Query 5 SPARQL Query for Identifying the Appropriate Sprinkler and Buzzer

```
1  PREFIX sta-lite: <http://140.115.110.71/sta-lite.ow1#>
2  PREFIX sosa: <https://www.w3.org/ns/sosa/>
3  PREFIX cigs: <https://140.115.110.71/cgis.owl#>
4  PREFIX indoorgml-lite: <https://140.115.110.71/indoorgml-lite.owl#>
5
6  Select DISTINCT ?Room ?Tasking_Sprinkler ?Tasking_buzzer
7  where {
8    ?Room sosa:hosts ?a.
9              FILTER regex(str(?a), "Device_TemperatureSensor_3")
10   ?Room cgis:isState ?Room_State.
11   ?edge_x indoorgml-lite:connects ?Room_State;
12             indoorgml-lite:connects ?connect_State.
13             FILTER (?Room_State != ?connect_State)
14   ?Thing_Sprinkler cgis:withinCellspaceOf ?Room_State;
15                    sta-lite:hasTaskingCapabilitiy ?Tasking_Sprinkler.
16   ?Tasking_Sprinkler sta-lite:description "Sprinkler control".
17   ?Thing_buzzer cgis:withinCellspaceOf ?connect_State;
18                 sta-lite:hasTaskingCapabilitiy ?Tasking_buzzer.
19   ?Thing_buzzer sta-lite:description "Buzzer control"
20 }
```

Query 6 SPARQL Query for Identifying the Appropriate Ventilation System and Buzzer

```
1  PREFIX sta-lite: <http://140.115.110.71/sta-lite.ow1#>
2  PREFIX sosa: <https://www.w3.org/ns/sosa/>
3  PREFIX cigs: <https://140.115.110.71/cgis.owl#>
4  PREFIX indoorgml-lite: <https://140.115.110.71/indoorgml-lite.owl#>
5
6  Select DISTINCT ?Room ?Tasking_Ventilation ?Tasking_buzzer
7  where {
8    ?Room sosa:hosts/sta-lite:hasDatastream/sta-lite:hasObservation
          ?Observation_smoke.
9          FILTER regex(str(?Observation_smoke),
     "Device_SmokeSensor_1_DS_ob_1")
10   ?Room cgis:isState ?Room_State.
11   ?edge_x indoorgml-lite:connects ?Room_State;
12             indoorgml-lite:connects ?connect_State.
13             FILTER (?Room_State != ?connect_State)
14   ?Tasking_Ventilation cgis:happensIn ?Room;
15                        sta-lite:description "ventilation control".
16   ?Thing_buzzer cgis:withinCellspaceOf ?connect_State;
17                 sta-lite:hasTaskingCapabilitiy ?Tasking_buzzer.
18   ?Thing_buzzer sta-lite:description "Buzzer control"
19 }
```

TABLE VII
RESULT FROM QUERY 5

Room	Tasking_Ventilation	Tasking_buzzer
R3-226	http://140.115.110.71/R3_STA.owl#Thing_R3_Building_Tasking	http://140.115.110.71/R3_STA.owl#Device_Buzzer_2_Tasking

TABLE VIII
RESULT FROM QUERY 6

Room	Tasking_Sprinkler	Tasking_buzzer
R3-226	http://140.115.110.71/R3_STA.owl#Device_Sprinkler_1_Tasking	http://140.115.110.71/R3_STA.owl#Device_Buzzer_3_Tasking

ventilation systems, and buzzers can be retrieved through SPARQL queries (Tables VII and VIII).

The simulated fire evacuation system demonstrates the capabilities of real-time environment monitoring and emergency response. Although navigation functionalities were not examined in the use case of the fire evacuation system, safe evacuation routes can be identified on the basis of IndoorGML route networks and sensor observations. By integrating city models and IoT resources, fire evacuation systems can efficiently identify the disaster status and effectively assist evacuation-related decision-making.

Overall, the use cases adopted in this study indicate that the proposed integration ontology provides a uniform and effective method for integrating city models and IoT resources. In the proposed integration ontology, data can be created independently and various views of `sta:Thing` can coexist in the integrated RDF. Thus, because the semantic Web technology is flexible, this ontology is capable of integrating IoT resources and city models in an interoperable manner for various smart city applications.

V. CONCLUSION AND FUTURE WORK

This work introduces an approach to effectively supporting and integrating interoperable inter–data set queries between IoT resources and open standard-based city models for smart city applications. Specifically, this research designed an integration ontology that characterizes the relationships between the SensorThings API, IndoorGML, and CityGML data models in an indoor space with consideration of different views of "things" in the IoT. This integration ontology can be used to integrate data from various resources for supporting different smart city applications.

To demonstrate the necessity and potential benefits of integrating IoT resources and city models, this research simulated two use cases, including a smart health care system and a fire evacuation system. Data from different resources can be retrieved through SPARQL queries through descriptions of the integration relationships in RDF format with the proposed ontology. The simulation results indicate that the proposed ontology successfully integrates IoT sensing and tasking capabilities with city models. Such integration can serve as an essential aspect of the cyber infrastructure of a smart city.

For achieving more comprehensive integration, future studies can incorporate additional semantic-rich open standards of city models and IoT resources, such as the BIM and W3C WoT standards, into the integration ontology. Because some relationships may be missing in the proposed integration ontology, additional relationships, such as geometrical relationships or relationships between "things," can be identified and described in future studies. Moreover, because the semantic Web technology is flexible and extensible, cross-domain integration can be efficiently, effectively, and continuously improved. The proposed ontology could also be presented to the OGC community for extending it as an OGC standard. The proposed integration ontology involves manual integration. Future studies can design approaches to automate this process. In addition, implementing the integration with new technologies, such as blockchain, fog computing, AI, would also be interesting discussions. Finally, the proposed ontology could serve as the foundation of a "digital twin" infrastructure to support real-world smart city applications and use cases.

REFERENCES

[1] M. Batty, "Big data, smart cities and city planning," *Dialogues Human Geogr.*, vol. 3, no. 3, pp. 274–279, Dec. 2013, doi: 10.1177/2043820613513390.

[2] R. Sánchez-Corcuera *et al.*, "Smart cities survey: Technologies, application domains and challenges for the cities of the future," *Int. J. Distrib. Sens. Netw.*, vol. 15, no. 6, Jun. 2019, Art. no. 1550147719853984, doi: 10.1177/1550147719853984.

[3] C. Franklin and P. Hane, "An introduction to geographic information systems: Linking maps to databases [and] maps for the rest of us: Affordable and fun," *Database*, vol. 15, no. 2, pp. 12–15, 1992.

[4] A. Gruen, "SMART cities: The need for spatial intelligence," *Geo-Spatial Inf. Sci.*, vol. 16, no. 1, pp. 3–6, Mar. 2013, doi: 10.1080/10095020.2013.772802.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014, doi: 10.1109/JIOT.2014.2306328.

[6] J. Carneiro, R. J. Rossetti, D. C. Silva, and E. Oliveira, "BIM, GIS, IoT, and AR/VR integration for smart maintenance and management of road networks: A review," presented at the IEEE Int. Smart Cities Conf. (ISC2), 2018.

[7] K. Stock and H. Guesgen, "Geospatial reasoning with open data," in *Automating Open Source Intelligence*. San Diego, CA, USA: Elsevier, 2016, pp. 171–204.

[8] C. Guney, "Rethinking GIS towards the vision of smart cities through CityGML," paper presented at the Int. Arch. Photogram. Remote Sens. Spat. Inf. Sci., 2016.

[9] C. Jing, M. Du, S. Li, and S. Liu, "Geospatial dashboards for monitoring smart city performance," *Sustainability*, vol. 11, no. 20, p. 5648, Oct. 2019, doi: 10.3390/su11205648.

[10] G. Gröger, T. H. Kolbe, C. Nagel, and K. H. Häfele, *OGC City Geography Markup Language (CityGML) Encoding Standard*, Open Geospatial Consortium Standard 12-019, 2012.

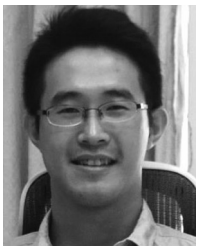
[11] J. Lee, K. J. Li, S. Zlatanova, T. H. Kolbe, C. Nagel, and T. Becker, *OGC Indoorgml*, Open Geospatial Consortium Standard 14-005r3, 2014.

[12] "LandXML, Non-Proprietary Data Standard for Land Professionals." LandXML. Org. 2000. [Online]. Available: http://www.landxml.org/about.aspx

[13] S. Azhar, A. Nadeem, J. Y. Mok, and B. H. Leung, "Building Information Modeling (BIM): A new paradigm for visual interactive modeling and simulation for construction projects," paper presented at the 1st Int. Conf. Construct. Develop. Countries, 2008.

[14] *Overview of Internet of Things*, Int. Telecommun. Union, Geneva, Switzerland, 2012.

[15] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou, and C. H. Lung, "Smart home: Integrating Internet of Things with Web services and cloud computing," presented at the IEEE 5th Int. Conf. Cloud Comput. Technol. Sci., 2013.
[16] L. Pescosolido, R. Berta, L. Scalise, G. M. Revel, A. De Gloria, and G. Orlandi, "An IoT-inspired cloud-based Web service architecture for e-Health applications," presented at the IEEE Int. Smart Cities Conf. (ISC2), 2016.
[17] A. Perallos, U. Hernandez-Jayo, E. Onieva, and I. J. G. Zuazola, and N. Aguiriano, *Intelligent Transport Systems: Technologies and Applications*. Chichester, U.K.: Wiley, 2015.
[18] F. Shrouf, J. Ordieres, and G. Miragliotta, "Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm," presented at the IEEE Int. Conf. Ind. Eng. Eng. Manag., 2014.
[19] S. K. Sowe, T. Kimata, M. Dong, and K. Zettsu, "Managing heterogeneous sensor data on a big data platform: IoT services for data-intensive science," presented at the IEEE 38th Int. Comput. Softw. Appl. Conf. Workshop, 2014.
[20] G. Xiao, J. Guo, L. D. Xu, and Z. Gong, "User interoperability with heterogeneous IoT devices through transformation," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1486–1496, May 2014, doi: 10.1109/TII.2014.2306772.
[21] P. Desai, A. Sheth, and P. Anantharam, "Semantic gateway as a service architecture for IoT interoperability," paper presented at the IEEE Int. Conf. Mobile Services, 2015.
[22] H. Wang, A. Gluhak, S. Meissner, and R. Tafazolli, "Integration of BIM and live sensing information to monitor building energy performance," paper presented at the CIB 30th Int. Conf. Appl. IT AEC Ind., 2013.
[23] J. Wang, H. Zhao, and S. Winter, "Integrating sensing, routing and timing for indoor evacuation," *Fire Safety J.*, vol. 78, pp. 111–121, Nov. 2015, doi: 10.1016/j.firesaf.2015.08.009.
[24] W. Zhu, A. Simons, S. Wursthorn, and A. Nichersu, "Integration of CityGML and air quality spatio-temporal data series via OGC SOS," paper presented at the Geospat. Sens. Webs Conf. (GSW), Munster, Germany, 2016.
[25] M. Sondheim, K. Gardels, and K. Buehler, "GIS interoperability," in *Geographical Information Systems*, vol. 2, P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, Eds. New York, NY, USA: Wiley, 1999, pp. 347–358.
[26] S. Liang, C. Y. Huang, and T. Khalafbeigi, *OGC SensorThings API Part 1: Sensing, Version 1.0*, Open Geospatial Consortium Standard 15-078r6, 2016.
[27] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic Web," *Sci. Amer.*, vol. 284, no. 5, pp. 28–37, 2001.
[28] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *Int. J. Human Comput. Stud.*, vol. 43, nos. 5–6, pp. 907–928, Nov. 1995, doi: 10.1006/ijhc.1995.1081.
[29] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquisit.*, vol. 5, no. 2, pp. 199–220, Jun. 1993, doi: 10.1006/knac.1993.1008.
[30] E. Miller, "An introduction to the resource description framework," *Bull. Amer. Soc. Inform. Sci. Technol.*, vol. 25, no. 1, pp. 15–19, Oct. 1998, doi: 10.1002/bult.105.
[31] Z. Bačić, T. Jogun, and I. Majić, "Integrated sensor systems for smart cities," *Tehnički Vjesnik*, vol. 25, no. 1, pp. 277–284, Feb. 2018, doi: 10.17559/TV-20160620125732.
[32] K. Chaturvedi, B. Willenborg, M. Sindram, and T. H. Kolbe, "Solar potential analysis and integration of the time-dependent simulation results for semantic 3D city models using dynamizers," paper presented at the 12th Int. 3D GeoInfo Conf., 2017.
[33] J. S. Kim, S. J. Yoo, and K. J. Li, "Integrating IndoorGML and CityGML for indoor space," paper presented at the Int. Symp. Web Wireless Geogr. Inform. Syst., 2014.
[34] S. Vilgertshofer, J. Amann, B. Willenborg, A. Borrmann, and T. H. Kolbe, "Linking BIM and GIS models in infrastructure by example of IFC and CityGML," in *Proc. Comput. Civil Eng.*, 2017, pp. 133–140.
[35] C.-L. Kuo and J.-H. Hong, "Interoperable cross-domain semantic and geospatial framework for automatic change detection," *Comput. Geosci.*, vol. 86, pp. 109–119, Oct. 2016, doi: 10.1016/j.cageo.2015.10.011.
[36] C. Peng and P. Goswami, "Meaningful integration of data from heterogeneous health services and home environment based on ontology," *Sensors*, vol. 19, no. 8, p. 1747, Apr. 2019, doi: 10.3390/s19081747.
[37] X. Liu, X. Wang, G. Wright, J. C. Cheng, X. Li, and R. Liu, "A state-of-the-art review on the integration of building information modeling (BIM) and geographic information system (GIS)," *ISPRS Int. J. Geo-Inf.*, vol. 6, no. 2, p. 53, Feb. 2017, doi: 10.3390/ijgi6020053.
[38] G. Gröger and L. Plümer, "CityGML—Interoperable semantic 3D city models," *ISPRS J. Photogram. Remote Sens.*, vol. 71, pp. 12–33, Jul. 2012, doi: 10.1016/j.isprsjprs.2012.04.004.
[39] H. K. Kang and K. J. Li, "A standard indoor spatial data model—OGC IndoorGML and implementation approaches," *ISPRS Int. J. Geo Inform.*, vol. 6, no. 4, p. 116, Apr. 2017, doi: 10.3390/ijgi6040116.
[40] S. Liang and T. Khalafbeigi, *OGC SensorThings API Part 2–Tasking Core, Version 1.0*, Open Geospatial Consortium Standard 17-079r1, 2019.
[41] C. Métral, R. Billen, A. F. Cutting-Decelle, and M. Van Ruymbeke, "Ontology-based approaches for improving the interoperability between 3D urban models," *J. Inf. Technol. Construct.*, vol. 15, pp. 169–184, Feb. 2010.
[42] V. Coors, "OGC 3D-IoT platform for smart cities engineering report," OGC Public Eng., Open Geospatial Consortium, Rockville, MD, USA, Rep. OGC 19-073r1, 2020.
[43] M. Kovatsch, R. Matsukura, M. Lagally, T. Kawaguchi, K. Toumura, and K. Kajimoto, *Web of Things (WoT) Architecture*, W3C Recommendation, Cambridge, MA, USA, 2020.
[44] *SmartM2M; SAREF Development Framework and Workflow, Streamlining the Development of SAREF and Its Extensions, V1.1.1*, ETSI Standard TS 103 673, 2020.



Chih-Yuan Huang received the B.S. and M.S. degrees in civil engineering and geoinformatics from National Central University (NCU), Taoyuan, Taiwan, in 2007 and 2008, respectively, and the Ph.D. degree in geomatics engineering from the University of Calgary, Calgary, AB, Canada, in 2014.

He is an Associate Professor with the Center for Space and Remote Sensing, NCU. His team is currently involved in the development of the Civil-IoT Taiwan Data Service Platform. His present research interests include interoperable cyber-infrastructure of the Internet of Things, digital twins, and artificial intelligence of things.

Dr. Huang is one of the editors of the Open Geospatial Consortium SensorThings API Part 1: Sensing Version 1.0 standard.

Yao-Hsin Chiang received the bachelor's degree from the Geomatics Department, National Cheng Kung University, Tainan, Taiwan, in 2017, and the double master's degrees in civil and environmental engineering from National Central University, Taoyuan, Taiwan, and Hiroshima University, Hiroshima, Japan, in 2020, respectively.

He is currently an Engineer with MOXA Inc., Taipei, Taiwan, for the industry automation connectivity technology.

Fuan Tsai (Member, IEEE) received the M.S. and Ph.D. degrees in civil and environmental engineering with concentration on remote sensing and spatial information from Cornell University, Ithaca NY, USA, in 1996 and 2000, respectively.

He is currently a Professor with the Center for Space and Remote Sensing Research, and the Department of Civil Engineering, National Central University, Taoyuan City, Taiwan. His research interests include remote sensing image processing and analysis, multimodal spatial analysis, digital city modeling, and smart city applications.