Enabling dynamic smart spaces using IoT-enhanced NGSI-LD data models

Nikolaos Papadakis, Georgios Bouloukakis, Kostas Magoutis

Enabling Dynamic Smart Spaces using IoT-enhanced NGSI-LD Data Models

Nikolaos Papadakis[1,3],Georgios Bouloukakis[2],Kostas Magoutis[1,3]
papadakni@ics.forth.gr,georgios.bouloukakis@telecom-sudparis.eu,magoutis@ics.forth.gr
[1]Institute of Computer Science (ICS), Foundation for Research and Technology - Hellas (FORTH)
[2]Télécom SudParis, Institut Polytechnique de Paris, France
[3]Computer Science Department, University of Crete, Greece

ABSTRACT

As cities continue to grow and evolve, so must their digital aspects as Internet of Things (IoT) applications and devices grow in number, usage, and ability to measure the world around us. Traditional ways of representing information, even virtually, have started to become obsolete, as there is a need to handle both the static and dynamic properties of real life environments. In this paper, we model the dynamic properties of smart buildings, their structural components, and data produced from IoT devices using the NGSI-LD protocol. To facilitate the modeling of existing buildings and further enhance them with IoT aspects, we introduce the *IFC2NGSI-LD parser* which converts existing IFC-based data models to NGSI-LD. The usefulness of our NGSI-LD models towards creating smart spaces, is demonstrated by executing complex IoT queries using the NGSI-LD API. Such queries can be applied in large-scale buildings, as enabled via the IFC2NGSI-LD parser.

KEYWORDS

IoT, Data modeling, Smart Buildings, IFC

ACM Reference Format:
Nikolaos Papadakis[1,3],Georgios Bouloukakis[2],Kostas Magoutis[1,3]. 2022. Enabling Dynamic Smart Spaces using IoT-enhanced NGSI-LD Data Models. In *Proceedings of the 3rd Connected World/Web Semantic Interoperability Workshop, November 7, 2022, Delft, Netherlands.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Smart Cities today embrace a variety of Internet and mobile technologies in order to improve the quality of life for citizens, transform the efficiency of public services, and generate new sources of growth for companies. Many businesses and organizations in smart cities have realized significant IT-based transformation processes in their daily operations in recent years [13]. As smart cities develop and evolve to include applications, such as energy management systems [6], the role and format of the virtual representation of buildings must evolve too in order to keep up with the current environment. Three-dimensional technologies are progressively displacing traditional paper-based and two-dimensional tools. These evolving IT-based information systems, also known as "Building Information Modeling" (BIM) technologies, provide integrated and collaborative building design, construction, and operation. Furthermore, the need for people to have detailed, effective, and constantly updated information about the status of buildings and their devices is becoming a must and requires more than the simple virtual representations of static properties of buildings provided by traditional BIM technologies. In order to provide the required information level and quality, there is an increasing need for considering and using virtual representations, i.e., *Digital Twin approaches*, that capture the most significant components of reality and therefore deliver a more realistic status of buildings and their devices.

The most common way to represent such information is through the use of graphs, with RDF ontologies on the one hand and property graphs on the other, with each having advantages and disadvantages inherent to their nature. There are many ontologies and data models related to smart buildings and their operations [1, 3, 16, 17]. Many of these are created with the modeling of static properties as their primary focus, and they do not consider how dynamic an environment can be. For instance, devices are coupled with their observations and a change in a physical device might severely impact the inner workings of a relevant IoT application. Stakeholders have often siloed systems, and, because of incompatibilities in modeling, communication across groups and subgroups is at times difficult. More often than not, they also primarily store this information in some form of virtual storage without being integrated into notification systems.

To tackle the above challenges, we use Next Generation Service Interfaces-Linked Data (NGSI-LD) as a flexible and dynamic basis for the modeling of smart spaces. In short, NGSI-LD is a data model for the standard representation of smart environments, and it provides an API for publishing, querying, and subscribing to context data. NGSI-LD has the unique characteristic of deriving its information model from property graphs, its meta model on the basis of RDF/RDFS/OWL, and partially on the basis of JSON-LD. Its purpose is to facilitate the open exchange and sharing of structured data among various parties. However, since NGSI-LD is a relatively new technology, existing openly published models for smart buildings are lacking, and no significant work has been done yet on the aspect of converting and mapping existing building data to NGSI-LD compatible models.

This paper presents an approach for the representation of dynamic smart spaces based on NGSI-LD. We first discuss the state-of-the-art in smart building modeling, which influenced the decisions made when creating our model (§2). We then present our NGSI-LD smart space model that captures the static properties of buildings, combined with a methodology and working parser for converting

CWSI'22, November 7, 2022, Delft, Netherlands

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

existing building data to our new representation (§3). In §4, we explain how we modeled the dynamic aspects of smart buildings, with the devices and their high level observations, and how we combined the static and dynamic property models to fully represent smart buildings. We conclude this work by validating these models, using the parser on open building data and by showcasing complex IoT queries that can be enabled with the use of the NGSI-LD on our models (§5).

2 BACKGROUND AND RELATED WORK

The easiest way to define BIM is as a set of IT tools used to create virtual representations of a building's whole physical and functional makeup. BIM models provide structural data but lack explanations of how the individual pieces of equipment and points work together. Beyond the simple representation of a building in virtual 3D format, there have been efforts to create richer representations of buildings that include dynamic aspects by using IoT devices and technologies. Graphs [15] are essential in analyzing such a wide range of datasets.

This section presents the Property Graphs (PGs) and the Resource Description Framework (RDF) graph-based models. Then, we provide background information for the NGSI-LD PG-based data model, which is used as the basis to introduce our *smart space models*. We conclude this section by comparing our proposed data model against existing ones.

2.1 Background

2.1.1 *Property Graphs and RDF.*

Both PGs and RDF triple stores offer ways to visually examine and represent related data (see Fig. 1). However, the two are significantly distinct from one another, and each has unique benefits in particular use-case scenarios. The goal of RDF is to provide a framework for publishing and exchanging structured data among a wide range of stakeholders, while PG was primarily created to store and query data as efficiently as possible. The implicit semi-formal data models that underlie the majority of modern graph databases are PG-based. They have received a lot of support, more so in business than in academia. They allow relationships to have characteristics attached to them (specified as key-value pairs), a feature that RDF does not directly support. Since RDF graphs code entity attributes as properties, they behave like actual graph arcs, but from the perspective of graph modeling. RDF graphs feature dead-end and meaningless arcs, making them fundamentally less effective than PGs as a means for storing information. However, PGs lack the formal foundation and standardization of RDF and cannot directly interact with linked data or other RDF datasets. Additionally, they are not well suited to common query languages like SPARQL or RDF-based reasoning tools.

2.1.2 *NGSI-LD.*

Many IoT implementations, from devices to data platforms, are detached from particular solutions. To overcome the interoperability gap between these installations, standardized data models were suggested. One of the suggested platforms, NGSI-LD, presents an information model and an API for standard management of context information and the exploitation of linked data. The JSON for

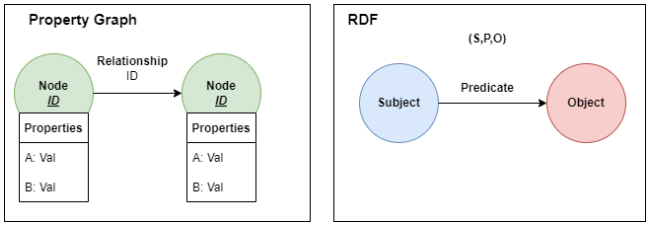

Figure 1: Property Graphs and RDF data models

Linked Data (JSON-LD) [14] serialization standard is the foundation of the NGSI-LD information model. JSON-LD design makes it easy to convert existing JSON to Linked Data with little changes. It is primarily intended for usage in Web-based programming environments, the development of interoperable Web services, and the storage of Linked Data in JSON-based storage engines. Instead of using RDF triples, which is typically used in Linked Data, JSON-LD serializes entity data in a simple and effective manner. This effectively utilizes the promise of semantics and connected open data. NGSI-LD is utilized in Smart Cities, Smart Industry, and Smart Agriculture, as well as the Internet of Things, Cyber-Physical Systems, Systems of Systems, and Digital Twins more generally.

The NGSI-LD information model [7] is derived from PGs. Entities, relationships, and properties are the key components of the NGSI-LD information model, as shown in Fig. 2. A real-world item, such as a building or a person, is represented by an entity. A relationship connects two or more entities, such as a person who works in a building. A property connects values to elements, such that it can identify that an entity corresponds to a real person. Due to its extensive data structure, it may be utilized for practically any data exchange situation throughout the life cycle of a building.

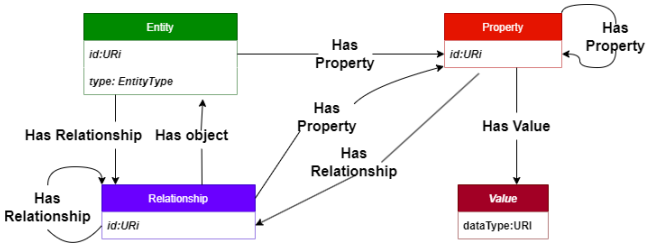

Figure 2: The NGSI-LD Information Model

2.2 Related Work

There have been many previous works that explore the modeling of smart building entities and their metadata. The Smart Appliances REFerence (SAREF) ontology [3] captures key features of connected and smart appliances. While SAREF does not cover the entire range of devices and sensors found in buildings, it can be mapped to the NGSI-LD Information Model, making use of this Context Information Management API specification for smart home applications. Similarly, the Building Topology Ontology (BOT) [1] and more complex metamodels such as the SemIoTic [16] which based on the popular SOSA/SSN ontology with some extensions, to represent relationships between the low-level IoT device world and semantic concepts, can also be mapped to NGSI-LD entities. The

Industry Foundation Classes (IFC) [2, 10] is a digital standard for describing the building asset domain. It supports vendor-neutral, or agnostic, and usable capabilities across a wide range of hardware devices, software platforms, and interfaces for many different use cases and is an open, international standard (ISO 16739-1:2018). It offers a comprehensive, standardized data format for the vendor-neutral interchange of digital building models that are a crucial foundation [10] for the development of Big Open BIM. In the work by Donkers et. al [12], they compared the two graph models (PG, RDF) both qualitatively and quantitatively using the Open Smart Home Dataset and a dataset that represented a kitchen. The comparison demonstrates that natively labeled PGs are less complex and perform better in complex graph traversals than atomic RDF. But RDF displays qualitative benefits for multi-domain and multi-stakeholder scenarios, including the usage of ontologies and HTTP URIs, making it a more reliable interoperability model.

Considering the above, we decided to use NGSI-LD as the base of our models. Whereas we decided not to map a specific ontology to NGSI-LD for our data representations, it will become apparent that existing ontologies and meta models such as the aforementioned, have been examined, and we do not disregard them but are in fact inspired by them in many ways. Using NGSI-LD does not limit us to just using property graph techniques and related technologies, as NGSI-LD entities can be easily imported into triple stores because they are serialized in JSON-LD. RDF reification [5], which supports any meta-data on them and permits the addition of properties to relationships, is the standard method for modeling property graphs. Reification is based on the automatic creation of blank RDF nodes across various RDF classes. All meta-data is attached to these blank nodes in order to take advantage of them. Thus, by using NGSI-LD as the base of our model, we try to have the best of both worlds.

3 NGSI-LD SMART SPACE MODELS

3.1 Modeling of Smart Buildings

The NGSI-LD community provides numerous data models, specifically the Smart Data Models project [4]. While there are many "official" data models provided by the NGSI-LD community as openly available information, a lot of them are still in the incubation stage and lack important elements. One such example was the data models related to the modeling of buildings. While buildings have been properly defined in great detail as entities with a generic model that can cover different building types, from farms to office buildings, the various sub-structural elements of a building have not been considered as a proper entity yet. The only entity that has been contemplated is the floor entity, which is still in an incubation stage. Since we wanted to have information that corresponded to smart buildings, we needed to extend this data model to include everything we considered missing and meet our needs. Obviously, not every building in a community will be "smart," and not every room will have devices, so the elements of a building must be modeled regardless of the presence or absence of sensors/actuators. Thus, we consider separating the modeling of smart buildings into the modeling of the structures and spaces and the modeling of the devices (sensors and actuators) present in a smart building.

3.2 Modeling of Spaces

The first thing is to decide what entities we want to be able to represent with our data model. It is surely important to have a separation of the various main components of a building to allow for easier querying of different entity types. There are a few clear-cut distinct elements we want to be able to depict, such as: (i) the building as a whole; (ii) the surrounding community where the building is located; (iii) the floors/The zones; and (iv) the rooms and their sub-elements (doors, windows, and stairs).

The digital depiction of those entities in our smart space data model needs to contain various relationships between the different entities. These relationships should allow for easy querying of important and distinct information. For example, imagine that we have a floor entity type and a building entity type. If only the relationship "*Building has Floors*" exists, having a floor with only its id and searching for what building that room belongs to would require searching all the buildings present in a broker and doing a content check of the "*Building has Floors*" relationship in order to find it. By establishing bidirectional relationships wherever logical and possible, such as "*Floor is in Building*", one can avoid unnecessary processing. The relationships of the previously identified smart building structural components can be seen in Fig. 3

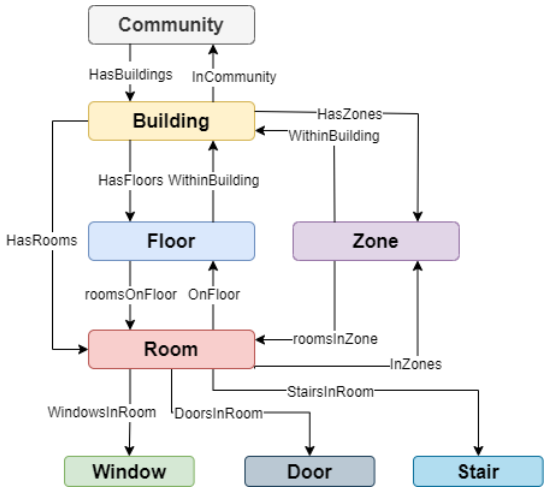

Figure 3: Relationships of building model entities.

Let us now see the various entities of our model in greater detail. First, we examine the community in which a building is located. The ability to categorize buildings in different communities for presentation and a more distinct division of the different structures makes it vital. A real smart community would have many more buildings than the one we have in the `HasBuildings` relationship in the example of Listing 1[1].

```
1  "id": "urn:ngsi-ld:Community:Test:SmartCitiesdomain:
       SmartBuildings:ExampleCommunity",
2  "type": "Community",
3  "HasBuildings": ["urn:ngsi-ld:Building:Test:SmartCitiesdomain
       :SmartBuildings:3isw_NcDz2ghLEYGeHmBHm"]
4  "location": {
5      "coordinates": [
```

[1]The GPS position features do not match reality because the listings presented in this section deliberately do not provide much real information.

```
        [
          [25.0750599,35.3070706],[25.0704902,35.3069743],[25.0705004,35.3041812],[25.0753909,35.3044001],[25.0750476,35.3070881],[25.0750599,35.3070706]
        ]
      ],
      "type": "Polygon"
    }
  },
  "name": "Test Building Community Area"
  ...
```

Listing 1: Community example

Then, we have the building entity[2] which as the name suggests is the general information about the entirety of a building. This model is derived from the building data model of the smart-data-models project, which we extended mainly by adding relationships with the various entities listed before (`InCommunity`, `HasRooms`, `HasZones`, `HasFloors`). A simplified example of a building with our NGSI-LD data model can be seen in Listing 2.

```
"id": "urn:ngsi-ld:Building:Test:SmartCitiesdomain:SmartBuildings:3isw_NcDz2ghLEYGeHmBHm",
"type": "Building",
"name":   "Default Building",
"category": ["civic"],
"dataProvider": "ICS_Forth",
"description": "A fake office building",
"floorsAboveGround": 3,
"floorsBelowGround": 1,
"InCommunity": ["urn:ngsi-ld:Community:Test:SmartCitiesdomain:SmartBuildings:ExampleCommunity"],
"HasFloors": ["urn:ngsi-ld:Floor:Test:SmartCitiesdomain:SmartBuildings:38vC2rMpPDpQ1cy52XqxrF",...],
"HasZones": ["urn:ngsi-ld:Zone:Test:SmartCitiesdomain:SmartBuildings:0aJ7egOIn66uiXCDk6uTq7",...],
"HasRooms": ["urn:ngsi-ld:Room:Test:SmartCitiesdomain:SmartBuildings:1OKccvw796O94cljFtqofp",...]
...
```

Listing 2: Building example

Next, we have floors and zones, which are brand new entities not present in the NGSI-LD data models repository. A Floor, as the name implies, is the entity that models each floor of a building. Its main relationships include the building the floor is contained in and the rooms the floor encompasses. Zones are areas within a floor or across multiple floors that belong in the same category. There are multiple zoning techniques. Normally, a house (a smart home) is not divided into zones, but because we are discussing smart buildings in general, an office building, for example, could have many zones, each of which could possibly correspond to offices of the same lab group, and so on. As with the floor entity, the zone entity has a relationship that points to what building it is in and a relationship that shows what rooms are included in each zone.

The room entity is particularly interesting and useful. Rooms are the smallest depiction of spaces in a building therefore, having information about every room in a building means having practically most of the building's information. Note that we consider hallways as rooms, and that rooms can be found in different zones. There can be a geometrical depiction of the shape of each component of a building entity. Since many buildings have a 3D representation, we permit rooms to have an attribute called "*relative position*" that covers both the 3D and 2D depiction of a room. We refer to it as such because each point in the geometry corresponds to a point relative to the coordinates of its surroundings rather than a GPS coordinate (in this case, relative to the building). The room has relationships with its sub-elements (`doors`,`stairs`,`windows`) and with the `floor` and `zone`(s) it belongs to. There is no need to have a relationship directly with the building it belongs to, as the 1-1 relationship `onFloor` also leads to a 1-1 relationship `WithinBuilding`. An example of a specific room can be seen in Listing 3.

```

"id": "urn:ngsi-ld:Room:Test:SmartCitiesdomain:SmartBuildings:1OKccvw796O94cljFtqofp",
"type": "Room",
"name": "K-3",
"onFloor": "urn:ngsi-ld:Floor:Test:SmartCitiesdomain:SmartBuildings:38vC2rMpPDpQ1cy52XqxrF",
"inZone": ["urn:ngsi-ld:Zone:Test:SmartCitiesdomain:SmartBuildings:0aJ7egOIn66uiXCDk6uTq7"],
"relativePosition": {
    "type": "Trimesh",
    "measurementUnit": "m",
    "Dimensions": "3D",
    "coordinates": [[0.35,9.68,-2.685],[0.35,9.68,-0.18000000000000016],[3.575,9.68,-2.685],[3.575,9.68,-0.18000000000000016],[3.575,9.075,-2.685],[3.575,9.075,-0.18000000000000016],[4.85,9.075,-2.685],[4.85,9.075,-0.18000000000000016],[4.85,6.055,-2.685],[4.85,6.055,-0.18000000000000016],[0.35,6.055,-2.685],[0.35,6.055,-0.18000000000000016]],
    "faces": [[2,0,1],[2,1,3],[4,2,3],[4,3,5],[6,4,5],[6,5,7],[8,6,7],[8,7,9],[10,8,9],[10,9,11],[0,10,11],[0,11,1],[4,6,8],[10,4,8],[0,2,4],[0,4,10],[9,7,5],[9,5,11],[5,3,1],[11,5,1]]
    } ,
"DoorsInRoom":["urn:ngsi-ld:Door:Test:SmartCitiesdomain:SmartBuildings:10ELxINu1AtQJwHiDxs4mK"] ,
"windowsInRoom":["urn:ngsi-ld:Window:Test:SmartCitiesdomain:SmartBuildings:1romkx9sb2mfYEttxEHWPc"] ,
"numberOfDoors": 1 ,
"numberOfWindows": 1 ,
...
```

Listing 3: Room example

Finally, we have the "smallest" entities, which correspond to various important entities present in rooms, such as windows, doors, and staircases. Their model mostly consists of their geometrical representation in space (either 2D or 3D).

The Building data models presented here are publicly available at https://github.com/SAMSGBLab/iotspaces-DataModels.

3.3 IFC2NGSI-LD parser

Following on from the previous subsection, we now have a way to properly represent smart buildings using NGSI-LD. However, there was a small caveat to our building modeling endeavors. *Do we want to manually create data models of existing buildings from scratch?* Obviously, this is very time-consuming and prone to human errors. For this reason, we leverage popular data models for buildings such as the **Industry Foundation Classes (IFC)** [2, 10] which is a digital standard for describing the building asset domain (§2). As shown in Fig. 4, it is a sophisticated data model that enables object-oriented representation of a building model's geometry and semantic structure. The structure is divided into its structural elements and its interior spaces, both of which are thoroughly delineated along with how they correlate to one another.

IFC is an open, international standard that supports vendor-neutral, or agnostic, and usable capabilities across a wide range of hardware devices, software platforms, and interfaces for many different use cases. Consequently, a solution to quickly generating our smart space data models from existing smart buildings was to develop a parser from IFC to our NGSI-LD data models. Handling directly IFC data from textual STEP physical files is not a trivial task, as it requires extensive knowledge of all the different entities

[2]https://github.com/SAMSGBLab/iotspaces-DataModels

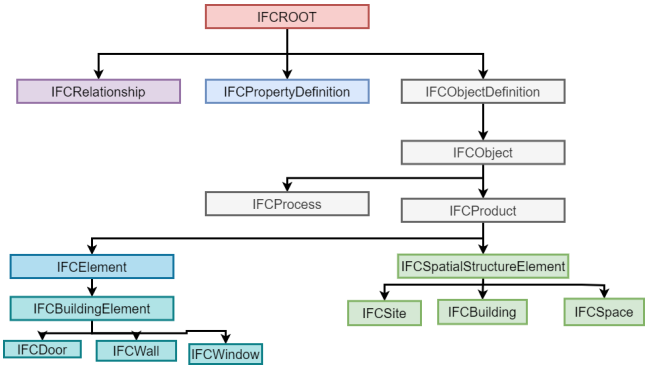

Figure 4: Some of the most important entities of the IFC hierachy

in the IFC schema, their relationships, and their hierarchies. We used the ifcOpenShell [3] library to process this format more easily. This is an open-source (LGPL) software library that makes it easier for users and software developers to work with the IFC file format. The biggest advantage of using this library is that IFC entities and relationships can be queried and retrieved by name, without requiring to search around a bunch of nested elements in the original text file. For example, to get the windows present in an IFC file, a simple "windows = ifc.by_type(`IfcWindows`)" function can be used. Skimming through the list of entities present in the IFC schema, one can match them to the entities present in our data model. This can be seen in Fig. 5

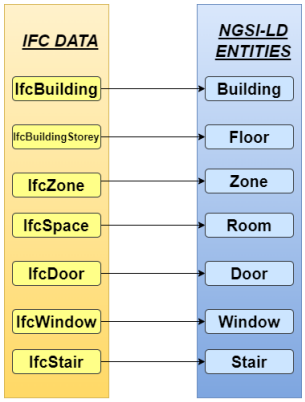

Figure 5: Mapping of .IFC data to ngsi-ld entities.

The inner workings of our parser can be described as follows: We query for every different entity that we are interested in Fig. 5 from the "largest" entity `Building` to the "small ones" (doors, windows, and stairs). Then, we create new NGSI-LD entities and, for each relationship between them or attribute they contain and finally, we create a matching NGSI-LD property (attribute or relationship) using the function that can be seen in Listing 4.

```
def create_ngsi_ld_attribute(Dictionary,Key,Value,
    Attribute_type):
    if(Value!='' and Value!=[]):
        if(Attribute_type=="Relationship"):
            Dictionary.update({Key: {"type":"Relationship","
    object":Value}})
        elif(Attribute_type=="Property"):
            Dictionary.update({Key: {"type":"Property","value":
    Value}})
        else:
            print("This is an error message")
```

Listing 4: Creation of NGSI-LD attributes from IFC attributes

Each different entity type is then "packaged" in the same file as depicted in Fig. 6 (e.g., all Room entities are written in a Rooms.ngsi-ld file.).

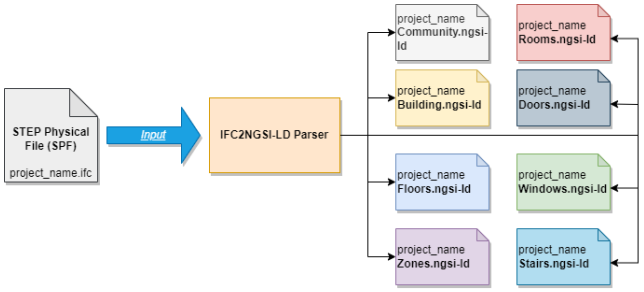

Figure 6: IFC2NGSI-LD parser

Because we created the data models to not be restricted by the IFC format, our data model does not contain all the information that may be included in an IFC file, and this currently leads to some information loss. This means that while our parser is capable of converting IFC to NGSI-LD files compatible with our data model, the reverse operation would lead to a more bare-bones IFC file than the original. A one-to-one mapping from IFC to a fully compliant NGSI-LD data model can be considered future work. The parser presented here is also openly available at https://github.com/SAMSGBLab/Iotspaces-IFC2NGSI-LD_parser.

4 MODELING OF DEVICES

Smart spaces (e.g., homes, buildings, etc.) include heterogeneous IoT devices (sensors/actuators) that vary in type and function [9], even between devices that produce the same data types. For example, an environmental humidity sensor from two different manufacturers may vary in physical characteristics and the technology used to send the information (LoRaWAN, MQTT). Humidity reading is the observed attribute of these 2 devices. Inspired by existing ontologies [1, 3, 16], there is a need to separate the physical device from the thing it measures or does in NGSI-LD based models. Developers will be able to create portable applications utilizing high-level observations rather than properties of particular devices by treating an observation of a device as an independent object. Multiple observations can also be associated with current or upcoming IoT devices. Thus, our model separates physical device properties from their observations or actuations. For this reason, we promote the observation/actuation attribute to a full-blown entity and create appropriate relationships with the device it originates from and the smart space it belongs to. This is a direct continuation of what we initially contemplated for the modeling of devices in our earlier

[3] http://www.ifcopenshell.org/

work [11]. While the initial decoupled design remains, this version of the modeling of devices is much more fleshed out.

Let us examine a specific example of a temperature sensor in a room shown in Fig 7. The physical sensor is the device that can be seen mounted on the wall of the room. The observation of the device is the temperature reading, and the space it belongs to is the room entity. Obviously, proper relationships need to be created between these entities. By including a camera in this room, we demonstrate how multiple device, observation, and smart space relationships are handled in Fig 8. Obviously, proper relationships need to be created between these entities. By including a camera in this room, we demonstrate how multiple device, observation, and smart space relationships are handled in Fig 8.

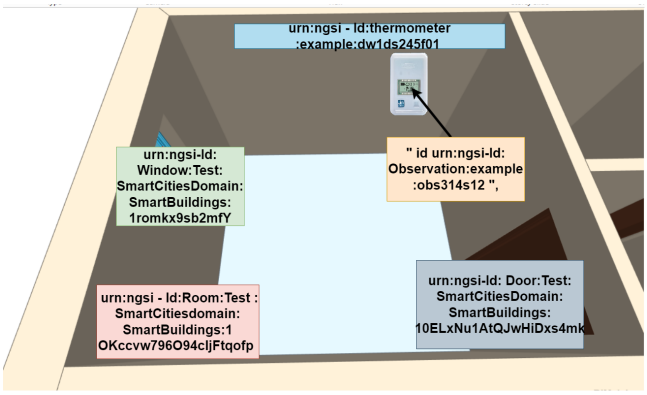

Figure 7: A temperature sensor on a room wall.

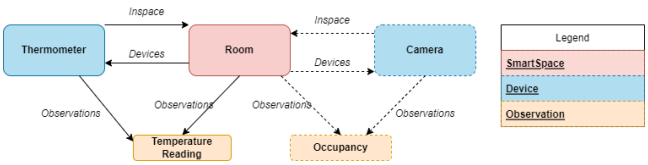

Figure 8: Diagram of room entities relationships.

First, the smart space in this example is the room. The room has a relationship called `Devices` that shows which physical devices are present in the room (`Thermometer`, `Camera`). The room also has the high-level observations that are derived from the devices. These are showcased with the `Observations` relationship; in this case (`Temperature Reading`, `Occupancy`). The devices have the `Inspace` relationship, which points to the smart space the devices are in, and, similarly to the room, the observation relationship. Our NGSI-LD model representations of this example can be seen in Listings 5, 6 & 7.

```
"id": "urn:ngsi-ld:thermometer:example:dw1ds245f01",
"type": "Thermometer",
"ControlledObservations": ["Temperature"],
"Observations": ["urn:ngsi-ld:Observation:example:obs314s12"],
"InSpace": "urn:ngsi-ld:Room:Test:SmartCitiesdomain:
     SmartBuildings:1OKccvw796094cljFtqofp"
....
```

Listing 5: Temperature Sensor Device

```
"id":"urn:ngsi-ld:Observation:example:obs314s12",
"type": "Observation",
"MeasurementType":"Temperature",
"Measurement": {"value":"34","Unit":"'C"}
"description": "Measurement of the temperature of a room",
"Coverage": "65%",
....
```

Listing 6: Temperature Measurement of room

```
"id": "urn:ngsi-ld:Room:Test:SmartCitiesdomain:SmartBuildings:1
     OKccvw796094cljFtqofp",
"type": "Room",
"name": "K-3",
"onFloor": "urn:ngsi-ld:Floor:Test:SmartCitiesdomain:
     SmartBuildings:38vC2rMpPDpQ1cy52XqxrF"
"inZone": "urn:ngsi-ld:Zone:Test:SmartCitiesdomain:
     SmartBuildings:0aJ7egOIn66uiXCDk6uTq7"
"DoorsInRoom": "urn:ngsi-ld:Door:Test:SmartCitiesdomain:
     SmartBuildings:10ELxINu1AtQJwHiDxs4mK"
"windowsInRoom": "urn:ngsi-ld:Window:Test:SmartCitiesdomain:
     SmartBuildings:1romkx9sb2mfYEttxEHWPc"
"numberOfDoors": 1,
"numberOfWindows": 1,
"Observations": ["urn:ngsi-ld:Observation:example:obs314s12","
     urn:ngsi-ld:Observation:example:occupancy7421wwa"],
"Devices":["urn:ngsi-ld:thermometer:example:dw1ds245f01","urn:
     ngsi-ld:camera:example:as31312d"]
...
```

Listing 7: Smart Space that houses the temperature sensor

As can be obviously deduced, there is no generic device model that covers every device but a dedicated model for each different device. For example, having a screen size attribute might be relevant for a smartphone device, but for a simple motion sensor, that would be a redundant attribute. In addition, we consider that not every device is just a sensor of sorts, some devices could be used as actuators. For instance, there could be a panel on a smart waste bin that lights up red when the bin is almost full, as depicted in Fig 9. All of the above, and specifically the various relationships present in the device data model, can be summarized in the following Fig 10. The observations the device makes and any applications that utilize these measurements will not be impacted by altering the physical device (which will probably lead to new device attributes).

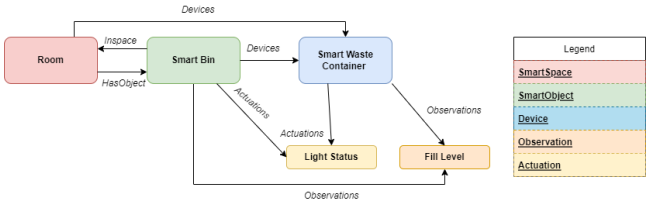

Figure 9: Example of smart waste bin inside a room.

5 APPLYING OUR SMART SPACE MODELS

This section presents validation tests that have been performed to ensure that our data model performs according to expectations as well as whether the parser accepts and produces correct information from IFC entities. First, we examine if potential IoT applications can perform complex queries on our models using the NGSI-LD API. In short, the various properties and relationships of the entities should allow for any possible query, simple or otherwise. Then, we examine if the parser works on existing large-scale IFC models by examining the produced data and performing the previously defined queries. We confirm that the parser produces correct and

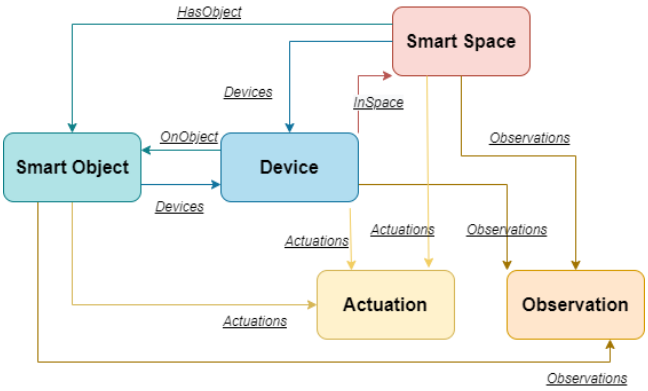

Figure 10: Relationship of devices in a smart space.

complete results from a variety of buildings and that the model combined with the API allows for complex IoT application queries.

5.1 Validating smart space models

We assume the existence of an administrative application for smart buildings that accepts user input for various parameters, for example, light and heating control commands. By its definition, the NGSI-LD API allows support for CRUD operations for buildings and devices, as well as data-driven list and search capabilities. We assume a building and IoT devices (temperature sensors, occupancy sensors, etc.), which we model using our building and device data models. The question here is, *can we, with the NGSI-LD API and our data models, provide the means necessary to perform complex queries that can handle the needs of such IoT applications without being very complicated?*

Imagine that the application has a dashboard that allows building administrators to continuously monitor the occupancy of each room. First of all, it would require that we find a way to get the list of the observations within the rooms through some kind of filtering, keeping only the observations that deal with occupancy information. Assuming we have information about one building, we simply subscribe to entities of type `Observation` with an attribute filter `MeasurementType==occupancy` and we only need the `Measurement` attribute to be modified and the id to directly interact with the specific measurement entity. If we had multiple buildings or wanted to monitor specific rooms, we would first need to perform a query to find the `Observation` Ids that relate to the rooms of interest using the `Observations` relationship, and then perform the previous filtered subscription on these entity ids.

The subscription is shown in the following Listing 8:

```
curl localhost:1026/ngsi-ld/v1/subscriptions -s -S -H 'Content-Type: application/ld+json' -d @- <<EOF
{
  "id": "urn:subscription:4",
  "type": "Subscription",
  "entities": [{
                "type": "Observation"
        }],
  "q": "MeasurementType==occupancy",
  "watchedAttributes": ["Measurement"],
        "notification": {
                "endpoint": {
                        "uri": "Application endpoint",
                        "accept": "application/json"
                }
        },
  "@context": ["https://raw.githubusercontent.com/SAMSGBLab/iotspaces-DataModels/main/device-models/context.json"]
}
EOF
```

Listing 8: Subscribe to changes in occupancy of rooms

Now that the application is aware of the occupancy of each room, a developer willing to have a good display of the information might have a render of the room. For more complex cases, imagine that we only want to render the room if the information we have is recent, say less than 24 hours ago. This can be done as follows: The first query, shown in Fig 11, is to get the initial information we want from the sub-elements of the room. The room has its 3D representation in the `RelativePosition` attribute, but to have a full render, we would also need its subentities (doors, windows, stairs) by using the appropriate relationships (`DoorsInRoom`, `WindowsInRoom`, `StairsInRoom`) and using them to get their own relative position attribute.

Figure 11: Example of variables used for the initial query using a Swagger interface.

Another application could utilize both contextual information and the capabilities of the NGSI-LD API to perform other relevant complex queries. For example, consider the following query: "*I want to find an emergency exit that is no further than 10 meters from my current location*". This can be done using geo-queries, the location of the door, and the information about whether a door is labeled for emergency use or not. An example of such a query is shown in Listing 9:

```

curl -X 'GET' \
  'http://localhost:1026/ngsi-ld/v1/entities/?type=Door&georel=near%3BmaxDistance%3D%3D10&geometry=Point&coordinates=%5Bmy%20current%20coordinates%5D&geoproperty=RelativePosition' \
  -H 'accept: application/json;application/ld+json' \
```

```
5    -H 'Link: <https://raw.githubusercontent.com/SAMSGBLab/
         iotspaces-DataModels/main/device-models/context.json>; rel
         ="http://www.w3.org/ns/json-ld#context"; type="application
         /ld+json"'
```

Listing 9: Query: Find at least one emergency exit that is no further than 10 meters from my current location

5.2 Validating the IFC2NGSI-LD parser

We validate our NGSI-LD parser using various IFC files and check if: a) it successfully creates the various NGSI-LD entities/files; b) the information extracted matches the initial data and there are no distortions; and c) the entities created are valid NGSI-LD data which can be inserted into an NGSI-LD compliant broker and queried accordingly.

The IFC data we used comes from the open IFC model repository [8]. The goal of this repository is to serve as a source for a wide range of well-examined IFC-based data files. The parser can process all those files and create the various entities without difficulty. To visually examine any possible distortions, we visualized the IFC using the BimVision viewer application [4]and we visualized the 3D information present in the converted NGSI-LD data using a simple Python 3D visualization library (trimesh library [5]). The comparison between the two visualizations can be seen in Fig 12.

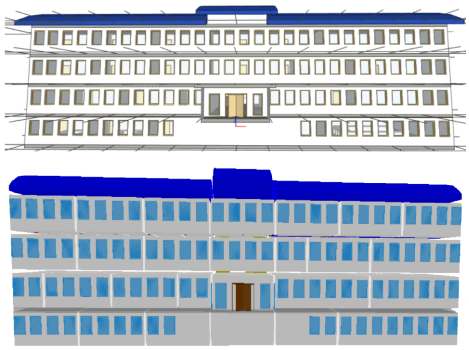

Figure 12: IFC building visualised using BimVision (above) vs. NGSI-LD visualization using python trimesh (below)

We tested the queries from the previous subsection using the converted datasets. For this, we used the NGSI-LD OrionLD broker [6], which implements most of the latest spec of the NGSI-LD API. Obviously, we could not examine the queries that involved devices without manually creating some devices and carefully embedding them into rooms using the appropriate relationships. The queries are functioning without problems, similar to the manually created test data, as expected. The data used for these tests can be found at https://github.com/SAMSGBLab/Iotspaces-IFC2NGSI-LD_parser.

6 CONCLUSION

In this paper, we present a way to model the static and dynamic properties of smart buildings based on the NGSI-LD Property Graph. We showcased how the NGSI-LD API can be harnessed to handle such information and perform complex IoT queries. A parser from the IFC file scheme to our data models has also been provided for the ease of generating compatible data from existing large-scale buildings. Overall, our work showcases the value of IoT-enhanced NGSI-LD Data Models in enabling dynamic smart spaces.

In the future, the limitations of current implementations of NGSI-LD compatible middleware can be explored and how the use of contextual information can be used to enhance the design and information processing of smart buildings.

[4]https://bimvision.eu/
[5]https://trimsh.org/trimesh.html
[6]https://github.com/FIWARE/context.Orion-LD

ACKNOWLEDGMENTS

We thankfully acknowledge support by the Greek Research Technology Development and Innovation Action "RESEARCH-CREATE-INNOVATE", Operational Programme on Competitiveness, Entrepreneurship and Innovation (ΕΠΑνΕΚ) 2014-2020, Grant T2ΕΔΚ-02848 (SmartCityBus). We also acknowledge our discussions with Stelios Stamatakis and his contribution to the search for ways of parsing .IFC data to our data model.

REFERENCES

[1] Building Topology Ontology (BOT). https://w3c-lbd-cg.github.io/bot/.
[2] Industry Foundation Classes (IFC). https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/.
[3] SAREF ontology. https://saref.etsi.org/.
[4] Smart Data Models project. https://smartdatamodels.org/.
[5] Ahmed Abid, Jieun Lee, Franck Le Gall, and JaeSeung Song. 2022. Toward Mapping an NGSI-LD Context Model on RDF Graph Approaches: A Comparison Study. *Sensors* 22, 13 (2022). https://doi.org/10.3390/s22134798
[6] Muhammad Saidu Aliero, Muhammad Asif, Imran Ghani, Muhammad Fermi Pasha, and Seung Ryul Jeong. 2022. Systematic Review Analysis on Smart Building: Challenges and Opportunities. *Sustainability* 14, 5 (2022). https://doi.org/10.3390/su14053009
[7] João Almeida, Jorge Silva, and Thaís Batista. 2020. A Linked Data-based Service for Integrating Heterogeneous Data Sources in Smart Cities. 205–212. https://doi.org/10.5220/0009422802050212
[8] Robert Amor and Johannes Dimyadi. 2010. AN OPEN REPOSITORY OF IFC DATA MODELS AND ANALYSES TO SUPPORT INTEROPERABILITY DEPLOYMENT. 16–18. https://doi.org/10.13140/2.1.1905.7609
[9] Nazmiye Balta-Ozkan, Rosemary Davidson, Martha Bicket, and Lorraine Whitmarsh. 2013. The development of smart homes market in the UK. *Energy* 60 (10 2013), 361–372. https://doi.org/10.1016/j.energy.2013.08.004
[10] Andre Borrmann, Jakob Beetz, Christian Koch, T. Liebich, and Sergej Muhic. 2018. *Industry Foundation Classes: A Standardized Data Model for the Vendor-Neutral Exchange of Digital Building Models.* 81–126. https://doi.org/10.1007/978-3-319-92862-3_5
[11] Georgios Bouloukakis, Chrysostomos Zeginis, Nikolaos Papadakis, Panagiotis Zervakis, Dimitris Plexousakis, and Kostas Magoutis. 2022. Enabling IoT-enhanced Transportation Systems using the NGSI Protocol. In *Proceedings of the 12th ACM International Conference on the Internet of Things* (Delft, Netherlands). https://doi.org/10.1145/3567445.3567460
[12] Alex Donkers, Dujuan Yang, and Nico Baken. 2020. Linked Data for Smart Homes: Comparing RDF and Labeled Property Graphs.
[13] Christoph Merschbrock and Bjørn Munkvold. 2012. A Research Review on Building Information Modeling in Construction—An Area Ripe for IS Research. *Communications of the Association for Information Systems* 31 (01 2012), 207–228. https://doi.org/10.17705/1CAIS.03110
[14] Manu Sporny, Gregg Kellogg, and Markus Lanthaler. 2014. JSON-LD 1.0 - A JSON-based Serialization for Linked Data. *W3C Recommendation* (01 2014).
[15] Dominik Tomaszuk. 2016. RDF Data in Property Graph Model. 104–115. https://doi.org/10.1007/978-3-319-49157-8_9
[16] Roberto Yus, Georgios Bouloukakis, Sharad Mehrotra, and Nalini Venkatasubramanian. 2019. Abstracting Interactions with IoT Devices Towards a Semantic Vision of Smart Spaces. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (New York, NY, USA) *(BuildSys '19)*. Association for Computing Machinery, New York, NY, USA, 91–100. https://doi.org/10.1145/3360322.3360859
[17] Roberto Yus, Georgios Bouloukakis, Sharad Mehrotra, and Nalini Venkatasubramanian. 2022. The SemIoTic Ecosystem: A Semantic Bridge between IoT Devices and Smart Spaces. *ACM Transactions on Internet Technology – TOIT* (2022).