Automation in Construction xxx (xxxx) 104129

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Review

IFC+: Towards the integration of IoT into early stages of building design

Angel Ruiz-Zafra [a, b, *], Kawtar Benghazi [a], Manuel Noguera [a]

[a] Department of Software Engineering, University of Granada, Spain
[b] Department of Computer Engineering, University of Cadiz, Spain

ARTICLE INFO

Keywords:
IFC +
Internet of things
Building information modelling (BIM)
Smart buildings
Industry foundation classes
IFC
Building design
Building models

ABSTRACT

The integration of Internet of Things (IoT) technologies into Building Information Modelling (BIM) provides significant opportunities to support useful services and functionalities for end users in buildings, enabling so-called smart spaces (*e.g.* smart buildings and smart homes). Current approaches perform this integration to access the deployed IoT devices once the building is constructed using Building Automation Systems (BAS), Building Information Systems (BIS) and Building Control Systems (BCS). Thus, the integration of IoT devices into the early stages of building design to describe IoT scenarios and IoT practices is still neglected, despite the benefits it can provide by being involved in the entire building lifecycle, *e.g.* cost estimation, stronger facility management, collaborative approach, simplification of tasks for developers/programmers through the integration with BIS, BAS and BCS, and a more holistic management of the IoT infrastructure.

This paper presents a solution to integrate IoT into the design stage (modelling) of buildings in BIM processes, through two main contributions: Industry Foundation Classes Plus (IFC+), an extension to the latest version of IFC where new types of entities have been added to support the modelling of IoT scenarios; and a digital object-based approach to transform smart-built environment specifications in IFC+ into consumable software, with the aim of supporting IoT scenarios. This paper illustrates the applicability of the proposed method in the context of design processes, the metamodel (schema) of IFC+, the proposed approach, and a proof of concept with a clarifying use case.

1. Introduction

The Internet of Things (IoT) is a trendy research topic owing to its widespread use and benefits [1–3], fostering its application in several diverse domains [4–6]. A most renowned IoT application domain is that of smart spaces (*e.g.* smart buildings and smart homes) [7], where several different solutions have emerged in recent years to provide new indoor functionalities [8,9].

Recent advances in IoT-related technologies, such as wireless sensors, data processing and Machine to Machine (M2M), and their convergence with the Building Information Modelling (BIM) paradigm, have the potential to transform the way of interacting with buildings and monitoring them, thereby enhancing the end-user experience.

BIM is a model-based process that provides the tools and insights to architecture, engineering and construction practitioners to plan, design, construct and manage buildings in an efficient manner [10,11]. In BIM, buildings and structures are represented using interoperable (*i.e.* shareable and exchangeable between different software tools) digital models that are created using software applications, such as AutoCAD®, Revit®, and SketchUp®. Some of the best known interoperable formats for such digital models are Industry Foundation Classes (IFC) and Construction Operations Building Information Exchange (COBie) [12].

IFC[1] is a complex and comprehensive standard for describing buildings, which depends on its own data model and corresponding metamodel to represent data. IFC specifications are formatted using the STEP ISO standard, either in eXtensible Markup Language (XML)[2] or in EXPRESS,[3] a data modelling language formalised in another ISO standard (*i.e.* ISO 10303). The IFC metamodel (*i.e.* IFC schema) defines several hundred entities to represent different building assets (*e.g.* door, space, and wall) and the relationships between them, such that most building types and their infrastructures can be modelled and represented with the IFC data model [13–16].

* Corresponding author at: Department of Software Engineering, University of Granada, Spain.
E-mail addresses: angelr@ugr.es (A. Ruiz-Zafra), benghazi@ugr.es (K. Benghazi), mnoguera@ugr.es (M. Noguera).

[1] https://standards.buildingsmart.org/IFC/
[2] https://www.iso.org/standard/40646.html
[3] https://www.iso.org/standard/38047.html

https://doi.org/10.1016/j.autcon.2022.104129
Received 29 July 2020; Received in revised form 30 November 2021; Accepted 3 January 2022
0926-5805/

In a smart building, where certain type of IoT infrastructure it is usually deployed, *things* (such as door, windows, and light) should be interconnected and aware of the current state of other elements because changes in the state of one thing in a smart environment may trigger the execution of several actions and services, possibly without needing human mediation. Thus, IoT connectivity must be planned from the onset. However, the basic set of IoT-related elements (*i.e.* sensors and actuators) that can be represented in IFC is not sufficient to represent several aspects related to smart building design, monitoring, and management [17–19].

In addition, several required features are not well-supported by the latest version of IFC (*i.e.* IFC4), such as the representation and query about the state of a piece of smart furniture (*e.g.* the state of a door or a light, simply for querying purposes or to change it), IoT networks and the interconnection between different nodes (*e.g.* servers, gateways, and other networks), or the authorisation of users to access certain parts of a building.

Moreover, IoT-based solutions employed in smart buildings are typically part of Building Automation Systems (BAS) [20], in which the building manager and her/his team use dedicated software to control the spaces, grant permissions, and monitor the accesses. IoT-based solutions are applied in a BIM process once the building is already constructed, in the context of building management processes [21].

Nevertheless, the support to IoT features from the early stages of building design has not been treated in depth in BIM, thus far. The integration of IoT-related functionalities in the design of buildings from their inception has several benefits, such as the documentation of smart building functionalities (*e.g.* building automation and management of devices, parking spot for employees, and motion detectors to conserve energy), thus enabling the specification of IoT ecosystems (sensors, networks, users and credentials) through Computer-Aided Design (CAD) software to address the design of an IoT scenario efficiently and iteratively as part of the entire building lifecycle process. This results in robust facility management processes and ultimately achieves better building environments [22].

The possibility of working with models is a very desirable characteristic as far as the support of IoT features in BIM processes is concerned, since each IoT infrastructure is managed by an underlying software. The use of models to represent IoT devices can helps to detect errors and inconsistencies in early stages of the software development, which in turn, has been repeatedly proved to reduce software lifecycle costs exponentially [23–25]. Similarly, there are starting to appear studies reflecting the positive economic impact of using BIM models in construction processes, although the lack of records and available data still difficulties to carry out these kinds of research [26,27].

This paper presents a set of mechanisms with the aim to address the concerns related to the integration of IoT into the early stages of building design and BIM process lifecycles. The two main contributions of this paper are *1)* an extension to the original IFC schema intended to support IoT features, that we have named IFC+, and *2)* a digital object (DO)-based approach to enable the automatic transformation of IFC+ entities into consumable software/services used by IoT applications and software for the management of buildings (*i.e.* BAS and Building Information Systems (BIS) software).

The work presented in this paper brings about important benefits, highlighting: *1)* the description of IoT scenarios in BIM from early stages in the building design through IFC+ and *2)* the automatic translation of IFC+ entities into ready-to-use software artefacts implemented as DOs and supported by the Handle System®, a comprehensive system for assigning, managing, and resolving persistent identifiers for digital objects and other resources on the Internet [28]. This automatic translation of IFC+ entities and DOs reduces development times; facilitates the integration of third-party software and minimizes errors in the access to consume the DOs, since service interoperable interfaces for IoT devices are automatically generated; and minimizes the potential mistakes derived from the human intervention in the development and deployment process (*e.g* wrong setting parameters). Likewise, the use of models to integrate the representation of IoT devices and construction elements in BIM models opens up the possibility to develop software that performs conformance and validity checks on such models, which results in improved model reliability. The software provided in the *Supplementary Section* allows the proposal to be tested using the appropriate tools to generate the IFC+ models from IFC ones (for example, exported from Revit) and transforms them automatically into a DO-object structure supported by the Handle System.[4] In addition, a Java-coded parser has been released ready-to-be used by developers or programmers to support IFC+ compatibility in customised software.

The paper begins by citing related studies on BIM, IoT, and the coexistence of both domains. It is followed by a description of the proposal that outlines the main components of our contribution. Subsequently, IFC+ is presented as well as the proposed approach including the different stages and elements required. Finally, a proof of concept (PoC) is described in detail, followed by the discussion and conclusions of the study.

2. Related work

The field of built environments is plagued by several diverse technologies that may be used in the different processes that may be a part of a BIM project, from CAD software for modelling buildings to sophisticated BAS and BIS to manage them [29].

Currently, the use of IoT-based technologies has fostered the development of applications to enhance buildings and their context, generating so-called smart environments or smart spaces, such as smart buildings and smart homes. However, research on the integration of BIM and IoT is still in its early stages, and most studies published thus far include concepts and theoretical approaches [21]. These studies focus on the integration of BIM and IoT from two different points of view or approaches: *1)* the use of IoT-based technologies to monitor buildings and support other aspects, such as construction logistics, and emergency response services; and *2)* the use of methods to integrate IoT and Information and Communication Technologies (ICT) together with BIM [30].

In the first approach, certain literature deals with BIM and IoT integration across several domains: construction, operation and monitoring, health and safety management, construction logistics and management, and facility management [30]. Several different projects presented for these domains propose solutions where IoT technologies are applied to monitor buildings within BIM processes. For instance, the projects presented in [31,32] to support real-time environmental monitoring through IoT technologies or the research described in [33,34] to provide emergency response services for buildings using IoT technologies and BIM.

In the second approach, the existing studies are oriented to investigate the methods to integrate IoT devices and BIM around three main concerns: *1)* contextual information to describe the building and IoT devices (*e.g.* sensors); *2)* light data storage to save time-series data from continuous sensor reading; and 3) the integration of both concerns, that is, contextual information and light data storage [30].

The projects in this second approach aim to use BIM technologies (*e.g.* CAD software and IFC data representation formats) and general-purpose technologies (*e.g.* databases, servers, and custom software) to integrate BIM and IoT. One example is the use of a particular CAD software (Revit) to transform the building contextual information into tuples of a relational database and store the time-series data gathered from IoT devices into a second database, both linked by a unique identifier [35,36]. Other projects propose the integration of contextual information and dynamic information (*i.e.* gathered periodically from IoT

[4] The Handle System must be installed. Please, check https://hdl.handle.net/20.1000/113 for more information about the installation process

devices) to provide useful functionalities (*e.g.* building heat maps and indoor positioning systems) through useful all-in solutions, such as the MITBIM platform presented in [37], or using query-oriented languages such as BIMQL [38]. Finally, most recent and sophisticated projects use semantic web technologies and ontologies, in which the contextual information (building context data) and dynamic information (*i.e.* the data gathered from IoT devices) are linked using specific languages (*e.g.* SPARQL) [39,40].

Reviewing one of the most complete surveys on this topic, presented by Tang et al. in [30], the integration of IoT into the early stages of building design (BIM) has not been addressed thus far, and most solutions focus on the integration of BIM and IoT in the use of IoT technologies to support data gathering once the building is constructed (*e.g.* monitoring, facilities, and services) or to link IoT-related technologies (*e.g.* devices) with static contextual information about the building.

Although certain approaches have been proposed to integrate BIM and ICT (*e.g.* to transform IFC specifications into relational database schemata and tuples), security and automation in data transformation still require further revision. Security is one of the main concerns in a smart building, that is, the grant of permissions, authorisations, and access to sensitive data. In addition, the transformation from IFC to a relational data model or any other data storage format requires additional software to handle the database, such as customised applications or Application Programming Interfaces (APIs) that must be implemented *ad-hoc*, which hinders the automation of the deployment of software related to smart buildings.

Many other technologies supported by buildingSMART organization[5] could be used instead of IFC to intend to support IoT from early stages of building design, such as another buildingSMART MVDs (Model View Definitions) like BAMie,[6] COBIe,[7] BIMSie, WALLie; and technologies such as BACNet or KNX [41]. However, we did decide to use IFC and our custom approach instead any other MVD and other technologies for the following reason:

- As IFC, the other MVDs mentioned are also oriented to the building construction processes, but likewise, with no support for IoT infrastructures and features. We decided to work with IFC, for being probably the most used/famous standard.
- Although there are many other technologies related to BIM that would seem suitable for the representation of IoT elements, such as the service framework BACnet and which enables the communication between electronic devices within a building, other main features related to IoT, such as access control, security issues and data integration are still missing.
- Regardless of the MVDs (IFC, BAMie, COBie, *etc.*) and the additional technologies used (KNX, BACnet), the goal is usually oriented to get a proper i) representation (MVD), ii) automation (*i.e.* KNX) and iii) management of building information. Usually, and according to buildingSMART, the integration of these three technologies could be done through the use of a BIM server[8] together with additional software (general purpose or *ad-hoc*, *i.e.*, customised). Thus, the deployment of an IoT-based scenario into a smart building requires many different technologies, skills and software. Our proposal reduces the number of technologies used and simplifies the integration of the different technologies (MVD, Services, Applications) by means of IFC+ to represent IoT-based smart spaces. The use of digital objects accessed through a friendly API to be consumed by programmers simplify development life cycles.

In this paper, IFC+ and a DO-based approach are introduced to address these two concerns. When deepening in the details of the aforementioned pieces of work, it is possible to realize that they do not accomplish a full integration of IoT technologies, particularly when it comes to the specification of design models. It can be said that they make use of IoT technology for some purpose, but IoT is not integrated into the BIM modelling artefacts. The IFC+ extension that we propose allows IoT devices and sensored spaces in a built environment (*e.g.* a temperature sensor, a light bulb actuator or the specification of secure-related parameters in an IoT network), to be treated as first-class entities, that is, to be modelled and represented the same as other construction elements (*e.g.*, walls, doors, stairs, *etc.*) in BIM projects. Thus, it is described as an extension to IFC that enables the modelling of buildings that support IoT-related features and security constraints.

In addition, the approach described in this paper enables the automatic translation of contextual information (IFC+) into an out-of-the-box solution to be used by IoT applications complying with security constraints. The approach, based on the use of DOs, enables the automation of software generation using model transformation techniques, with the aim to represent IoT-related building assets and a common way to consume software related to the IoT infrastructure into the building [42]. With the approach proposed it is possible, in an automatic way, generate an entire software infrastructure (supported by DOs) from the CAD model.

This software infrastructure contains consumed software elements (DOs) that can be accessed through an ease-to-use REST API, allowing the consumption of the DOs by third-party applications. Thus, buildings constructed using an approach similar as it is proposed will share the same syntax of the REST API, which significantly foster the communication, interconnection and integration of buildings, in the context of smart cities.

3. Integrating IoT into building design environments

The IFC data model was designed to describe buildings and construction data, with IFC4 being the latest version of the standard [13,14]. This technology allows building supplies, furniture (such as doors, tables, and windows), and building infrastructures (*e.g.* electrical and pipeline installation) to be represented. The IFC schema is defined using EXPRESS, a data modelling language for product data based on STEP (ISO 10303) [43]. Thus, the IFC4 schema defines entities to describe doors, windows, walls, and any type of furniture as well as relationships between them.

However, regarding IoT, several elements or features are missing in IFC4 which are required to design smart spaces that use IoT technologies. For instance, the entity *Door*, represented in IFC4 as *IfcDoor* or *IfcDoorStandardCase* (type and subtype respectively), has a set of attributes such as color, size, and description, but is unrelated to the current state of the door (*i.e.* open/closed); hence, it is not possible to devise mechanisms to monitor such state using IFC. The schema specification for sensors in IFC is another paradigmatic example of this situation. The concept of sensor is also supported in IFC4, as the entity *IfcSensor*, but no information is provided on the way to interact with each sensor (*e.g.* communication protocol, what information is provided, and data structure), and thus, the potential of IoT is not fully leveraged.

In this section, IFC+, an extension to IFC4, is presented to support IoT features, with the aim of providing architects/designers, the possibility of modelling IoT scenarios at the early stages of building design, in conjunction with CAD software. For instance, when setting up the sensors in the lounge of a building, it would be desirable to additionally specify the authorised users to access these sensors, the IoT-based networks to use, and the configuration parameters required in the different elements involved in communication with such sensors (*e.g.* gateways and servers).

[5] https://www.buildingsmart.org/
[6] http://docs.buildingsmartalliance.org/MVD_BAMIE/
[7] http://docs.buildingsmartalliance.org/MVD_COBIE/
[8] https://github.com/opensourceBIM/BIMserver

IFC+ is defined as a schema and has been modelled following the same syntax and philosophy: a hierarchical and well defined structure of entities (tree data structure). In IFC+, the main entity is IfcPlusIoT, which is a sub-entity of IfcObject, a core entity of IFC, and all entities defined in IFC+ inherit from IfcPlusIoT or any of its sub-entities. In this way, as for the description of IoT elements, IFC+ has the same reliability as IFC to model buildings and IoT scenarios.

Due to IfcObject *"is the generalization of any semantically treated thing or process"*,[9] IFC+ entity also inherits (for transitivity) from the most general entity in IFC and thus, any IFC+ entity will also have the general attributes that any IFC entity has. In this way, we enable the definition of our custom entities (*i.e.,* IfcPlusXXX) at the same level as any other thing in IFC, such as a wall or a door. This entails that, in terms of software processing, an object like an IoT-based badge reader or any other kind of IoT device is treated the same in IFC+ as a door or any other construction element from the IFC standard.

This inheritance also entails two main benefits:

1. The presence of required attributes that must be specified for any IFC entity, also in IFC+ entities. These attributes are also inherited from other entities (IfcRoot, IfcObjectDefinition) like the GlobalId, name, owner, description, relationships (associations) and type of object. This will help to ensure that custom IFC+ entities share the common data structure of IFC entities, which helps in the interpretation, management and processing of IFC+ files by CAD software or custom software in the future by means of homogeneous representations.
2. Since IfcPlusIoT inherits from IfcObject, and in IFC+ any entity inherits from IfcPlusIoT, it is possible to link together original IFC entities and IFC+ entities. This feature is mandatory so as to enable CAD software to represent IoT-based assets, such as smart doors, linking IFC entities (*e.g.*, a door) with IoT devices (*e.g.,* an RFID card).

Finally, we propose a novel approach based on DOs to generate a ready-to-use consumable software from IFC+ specifications to enhance the automation of smart spaces. Both contributions are described in this section.

3.1. IFC+ genesis

The formalisation of IFC+ to describe IoT scenarios in the early stages of building design was conducted in two different stages—requirement analysis and design—to detect the required IoT-related features and an appropriate way to define them to comply with the current IFC specification and schema.

3.1.1. Requirements analysis

To design a suitable IFC+ schema, an in-depth requirement analysis was conducted to detect new elements and features required to describe IoT scenarios. This requirement analysis did start with periodical meetings hold in the scope of PETRAS project (https://petras-iot.org/), with the staff of the BRE (Building Research Establishment) in Watford (UK), researchers from Newcastle University (Urban Sciences Building), researchers from Digital Catapult Ltd., and people from The Bartlett School of Architecture (University College London) from November 2016 to January 2018. These meetings represented a first benchmark for the elicitation of the desired features that an extension to IFC that would include support for IoT infrastructure design, should have. The requirement analysis was completed after a deep review of the literature, with the aim to define the main requirements of IoT ecosystems that are not included in IFC, such as the works presented in [7,22,44].

[9] https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC1/HTML/schema/ifckernel/lexical/ifcobject.htm

Such analysis revealed the following important concerns to be addressed: *1)* things in buildings, *2)* state of things, *3)* things locations, *4)* interaction between things and networks, and *5)* security.

3.1.1.1. Things in buildings. The IoT is a large domain with a wide range of processes, interconnected devices, and supporting technologies. In an IoT scenario, there are things, that is, well-identified items with interfaces to interact with them through a communication protocol technology (*e.g.* network) [45].

However, not all elements inside a building should be considered as *things* within an IoT scenario. For instance, a specific wall might not be relevant from an IoT perspective. In contrast, a smart door (*i.e.* a door equipped with sensors, actuators, and RFID tags) must be considered as a thing to check whether the door is open or closed, obtain the list of authorised users to open it, and manage/monitor it remotely. Thus, in a building modelled by IFC4, certain entities will be relevant in an IoT scenario, whereas others will be transparent.

3.1.1.2. State of things. Generally, things in an IoT scenario may be in different states (*e.g.* a door opened, a window closed, an air conditioning machine off, and a lamp on). The definition of the state of the different things is not supported natively in IFC, although it is possible to define and link custom properties through the *IfcPropertySingleValue* entity.

However, these custom properties are tailored and defined *ad-hoc* by each architect/designer. Thus, the lack of standardisation and reference schema often leads to situations in which two or more models aim to represent the same state of an entity in different ways. For instance, the first model could represent a particular state using an attribute called state that could have an integer number to represent the different states (*e.g.* -1, 0, 1, 234, and – 123), whereas the second model could use an attribute called *position* that represents the same state using a character string (*i.e.* a word or small piece of text, such as *on*, *battery_low*, *off*, and *shut_down*). Applications using any of these two schemata cannot interact or share information at all because they process states in a particular manner and expect information to be codified in a particular manner, rendering it unfeasible to foresee and process all possible ways of representing the state of an entity. This fact hinders the integration of different schemata into the same system. Hence, it is necessary to unify the representation of common attributes and features present in an IoT-based smart space.

3.1.1.3. Things locations. In IFC, the place of an entity is relative to another entity, generally the entity where it is contained. For instance, to identify the location of a door (an *IfcDoor* entity), the wall (an *IfcWall* entity) where it is placed and the floor of the building where the door is located (an *IfcSpace*) must be identified. In terms of smart spaces, it would be desirable to describe the location of things in a building in absolute terms to support functionalities required in IoT settings, such as 'turn off the lights of the first floor' or 'open all the doors'.

3.1.1.4. Interaction with the things and network. An in-demand feature of things in IoT settings is the capability to be accessed by other things, applications, processes, and users.

IFC entities are described using common attributes, such as name, description, size, and color. Although new entities have been added to represent IoT-related elements, the attributes required to describe how to interact with them; such as communication protocol, address, and interface; how the device sends the information; and what information is sent by an IoT device are still missing. Therefore, support is required to describe additional entities to represent interactions between elements present in an IoT setting.

Furthermore, a crucial element to support these interactions is the network. Similar to an electrical or pipeline installation in a building,

smart spaces deploy different types of networks to provide communication channels to the BAS and BIS. Smart spaces typically have a specific network technology, such as wireless sensor networks (WSNs) or low power wide area networks (LPWANs), to transmit data in IoT settings [46]. These types of IoT-oriented networks are not supported in the current IFC standard but are required to define the interactions between elements. Hence, they must be included to specify a fundamental part of IoT settings.

3.1.1.5. Security. Security is one of the main concerns to be addressed in an IoT design [47]. In a BIM–IoT solution, it is necessary to tag the different things in a building and ensure that only authorised and authenticated users access them. For instance, in a smart space with different types of users (*e.g.* building manager, office worker, and maintenance worker) and different types of things (*e.g.* smart doors, lights, and sensors), certain users will be authorised to manage the entire smart space (*e.g.* a building manager) or part of it (*e.g.* an office worker or a maintenance worker).

These authorisation and authentication features between things and users are not supported in IFC; however, they are supported by the BAS and BIS once the building is constructed. Thus far, it is not possible to describe these security constraints in the early stages of building design.

3.1.2. IFC+ design

According to the different concerns described in the previous section, and as an extension to IFC (*i.e.* to its latest version, IFC4), new entities, attributes, relationships and types have been defined in IFC+.

The IFC schema has a well-defined structure. Hence, the new elements of IFC+ were defined to be compliant with the specifications of IFC. IFC+ metamodel (Fig. 1) is the general description, containing all the possible elements to be used. The different files (*ifcplus* extension) used to describe buildings, also called instances, are created from the schema. An essential technology to support this generation of instances from the original schema is the use transformation models technology [42].The use of models and transformation models automate part of the implementation, but also ensure that IFC+ files are consistent with respect to the original schema, because it is not possible to construct an instance with entities or elements that are not represented in the original schema. Furthermore, model transformation techniques along with the description of IoT elements can help to detect inconsistencies and problems in early stages of the building construction process, in particular as far as interaction software between IoT devices and applications is concerned. This is a source of software overcost widely studied and proven in the literature [48].

In IFC, the names of all elements begin with the word *Ifc*, followed by the name of the entity (*e.g. IfcDoor*, *IfcWall*, and *IfcWindow*). To differentiate the entities that have been added in IFC+ from the existing ones in IFC, we adopted a convention to begin IFC+ entities with the term *IfcPlus*, followed by the name of the entity (*e.g. IfcPlusSensorSecurity* and *IfcPlusNetwork*).

IFC+ had one root entity, *IfcPlusIoT*, being positioned at the same level as other upper-tier entities in IFC (*IfcActor*, *IfcProcess*, and *IfcProduct*) and inherited directly from *IfcObject*, which is the generalization of any semantically treated entity or process in IFC. Thus, *IfcPlusIoT* inherited the attributes of *IfcObject* that provided crucial features, such as the attribute *IsDeclaredBy*, which enabled the association of certain entities with others, allowing the linking between IFC+ and IFC entities. Furthermore, this inheritance provided attributes such as name, description, and unique identifier (global identifier).

Moreover, the definition of *IfcPlusIoT* as the root IoT element below the *IfcObject* entity permitted the addition of new elements when necessary, preserving the original structure of IFC and ensuring backward compatibility and consistency between the new elements added in IFC+ and the existing ones in IFC.

The entities added in IFC+ to describe IoT scenarios are organised, conceptually, into four main topics, according to the concerns detected in the *Requirements analysis stage* (Section 3.1.1). The topics are as follows:

1. *Network*. In smart spaces, several devices (things) are interconnected or connected to external actors (*e.g.* users, software, and devices) to exchange data. IoT entities belong to a network that must be defined and represented in IFC+.
2. *State*. Supplies and furniture in a building have state, but are not represented natively in IFC, such as the state of doors, windows, and lamps. Therefore, the representation of the state of IoT entities was essential in IFC+ to support IoT settings.
3. *Location*. Although, in IFC, it was possible to calculate the location of an entity by recursively calculating the location of the entities where it was contained until reaching the root *IfcProject* container, IFC+ provided additional entities to describe the location of a thing more efficiently and easily.

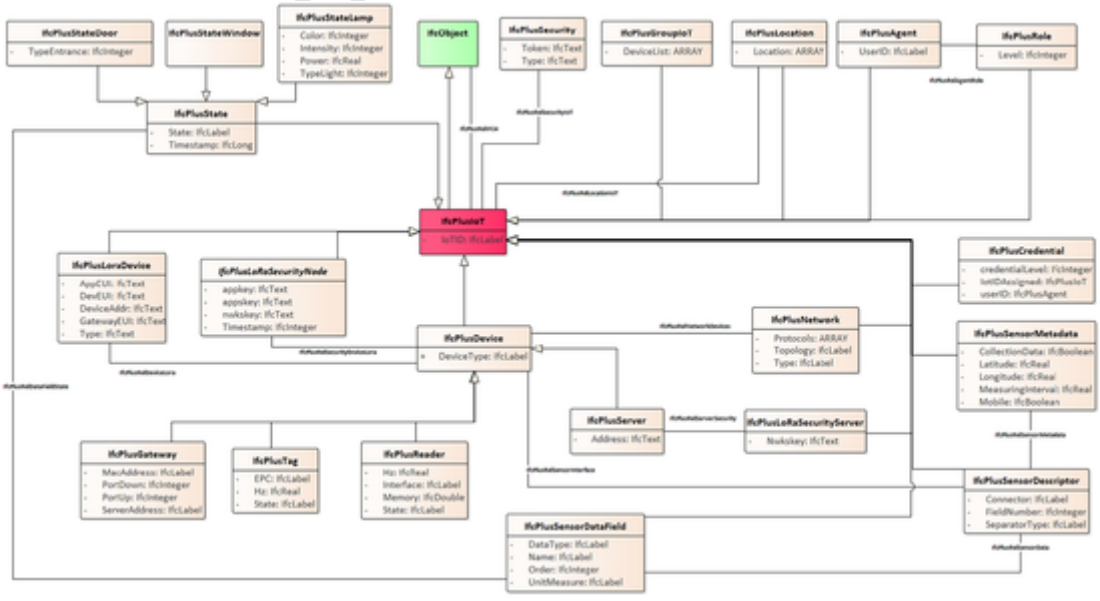


Fig. 1. IFC+ metamodel (or *schema* in IFC jargon).

4. *Security*. Security is a crucial concern in any IoT scenario. Hence, it was addressed and covered in IFC+ to secure the authentication of the users and the integrity of the data. Therefore, additional entities were added to support the description of secure IoT scenarios in smart spaces.

The casuistry for the representation of IoT scenarios is really significant and constantly growing. Thus, in IFC+, several entities to represent generic IoT elements were added, such as devices (*IfcPlusDevice*), networks (*IfcPlusNetwork*), states (*IfcPlusState*), and security (*IfcPlusSecurity* and *IfcPlusCredential*). Below each generic element, there were several entities that represented a required and specific concern in built environments. For instance, the representation of an integrated circuit card based on the use of smart cards and badge readers, commonly used for smart doors in smart buildings, could be represented in IFC+ by the *IfcPlusTag* and *IfcPlusReader* entities, respectively.

An illustrative example is that of networks for IoT applications. Currently, several IoT networks, such as Long Power Wide Area Network (LPWAN), have different features which are defined and maintained by either public or private organisations. LoRaWAN is a LPWAN that enables long-range Internet transmissions (of more than 10 km) [49], allowing the deployment of Wireless Sensor Networks (WSNs), both indoors and outdoors, and making it a suitable choice for smart-built environments [50,51]. In IFC+, several entities permitted the complete representation of a LoRaWAN network and its configuration parameters, supporting different end nodes, such as IoT devices, (*IfcPlusLoRaDevice*), LoRa gateway (*IfcPlusGateway*), Lora server (*IfcPlusServer*), and security-related parameters (*IfcPlusLoRaSecurityServer, and IfcPlusLoRaSecurityNode*).

Another example to illustrate the capability of IFC+ is related to sensors and actuators. Sensors and actuators are devices commonly present in IoT applications and smart environments (*e.g.* temperature sensors, humidity sensors, and actuators to open a door). These types of devices were represented in IFC+ as entities of the type *IfcPlusDevice*. Although *IfcSensor* and *IfcActuator* entities existed in the original IFC schema, the representation of sensors/actuators for smart spaces were enhanced in IFC+ by a set of additional entities to specify metadata for the sensor (*IfcPlusSensorMetadata*) as well as to specify the interaction with the device. Because the interaction with a sensor/actuator is a complex programming task, two entities, *IfcPlusSensorDescriptor* and *IfcPlusSensorDataField*, were added in IFC+ to describe the technology used and thus, enhance the integration of the sensor/actuator in the smart-built environment, for example, by using third-party description languages that are easily compatible with IFC+, such as Sensor Markup Language (SenML), Sensor Model Language (SensorML), or other customised model-based approaches [52,53].

Fig. 1 presents the IFC+ metamodel (schema) described in the Unified Modelling Language (UML) with the entities added to the IFC schema and their relationship with each other. The complete list of added entities and a short description of each one are provided in Table A.1, in Appendix A.

The complete IFC+ schema can be obtained from the link provided in the *Supplementary Materials* section.[10]

3.2. A DO-based approach to enable the use of IFC+ entities by IoT applications

Although IFC+ allows the modelling of IoT scenarios, once the building is constructed, IoT applications must be implemented to support the interactions. These applications will be configured and will interact with the devices, networks, and elements specified in the IFC+ file of the building. To simplify the development of these applications and the management of the different IoT-related elements described in IFC+, we propose a solution that automates the generation of software artefacts from the specifications of IFC+ entities. These software artefacts, represented as DOs [54], could be used by customised *ad-hoc* IoT applications or by existing commercial BAS, BMS, or BIS.

IFC+, as an extension to IFC, is a schema to support and model buildings with IoT features.

A *DO* is a data record that contains data, state and information about one entity in a domain [54]. Typically, DOs are used as a digital representation of physical or digital elements and their features. A specific implementation of DOs is supported by the Handle System®, an open-source and comprehensive system for assigning, managing, and resolving persistent identifiers for DOs over the Internet [28].

In the Handle System, everything is a DO, which is identified by a unique identifier composed of two different parts: a *prefix* (generally represented by a number) and a unique alphanumeric name called *suffix*; both separated by the character '/', that is, *prefix/suffix* (*e.g. 55,555/device1*). These DOs can be consumed through a Representational state transfer API (REST API), provided natively by the Handle System.

In terms of security, the Handle System guarantees the access and management of DOs, providing easy-to-use and well-tested security mechanisms for authentication and authorisation. IFC+ is well-aligned with these features. In the Handle System, the authentication is the process to be identified as a DO (that is, similar to being identified as a user in any other system), whereas the authorisation is the possibility to access and/or modify a specific DO. Easy-to-use and well-tested security mechanisms for authentication and authorisation are implemented and supported by the Handle System, such as basic authentication schema (user id and password) and digital certificates for authentication and the use of Access Control Lists (ACLs) for authorisation [54].

Finally, the Handle System supports a naming system similar to Domain Name System (DNS), which can be used to represent and resolve DOs with a hierarchical structure.

To expedite the development and deployment of IoT scenarios in buildings modelled by IFC+, we propose the transformation of IFC+ entities into DOs supported by the Handle System. This entails representing each IoT-related element described in IFC+ as a DO with *1)* a unique identifier, *2)* a custom data structure to represent the entity information and *3)* security features, assuring security features of the building that are described in the IFC+ model through the corresponding entities (*i.e. IfcPlusCredential* and *IfcPlusSecurity*). Fig. 2 depicts the proposed approach.

In this approach, the architect designs (*Stage 1*) the building using a specialised CAD software that supports the IFC+ schema. Consequently, an IFC+ file with a model of the building is generated.

Using the model of the building and a dedicated software, the different IoT-related entities described in the IFC+ file are transformed into DOs in the Handle System, producing a hierarchical tree-like entity structure to represent the entire building (*Stage 2*), as illustrated in Fig. 3.

In addition to the features provided by the Handle System (*i.e.* security mechanisms, ACLs, and metadata about each entity), the DOs have a data structure to represent the information and features of the IFC+ entities. This data structure is managed by the Handle System as a plain text field, but it could be enhanced for built-oriented goals, such as the proposal described in [55], consisting of enhanced data structures for DOs called *templates*, which enable *1)* the description of complex data structures through a hierarchical organization and *2)* the linking between DOs' data structures.

The IFC+ file is only used to describe the IoT scenario itself and the information required to configure it, which is transformed into Digital Objects in this stage. In this way, additional data such as values gathered from sensors, networking traffic logs, *etc.* is not stored, in any way, in the IFC+ file. This data must be handle (implemented) by the system developers and analysts in Stage 5, who decide how to handle them properly (storage, organization, analysis, purpose). This DO-based ap-

[10] https://github.com/bihut/ifcplus

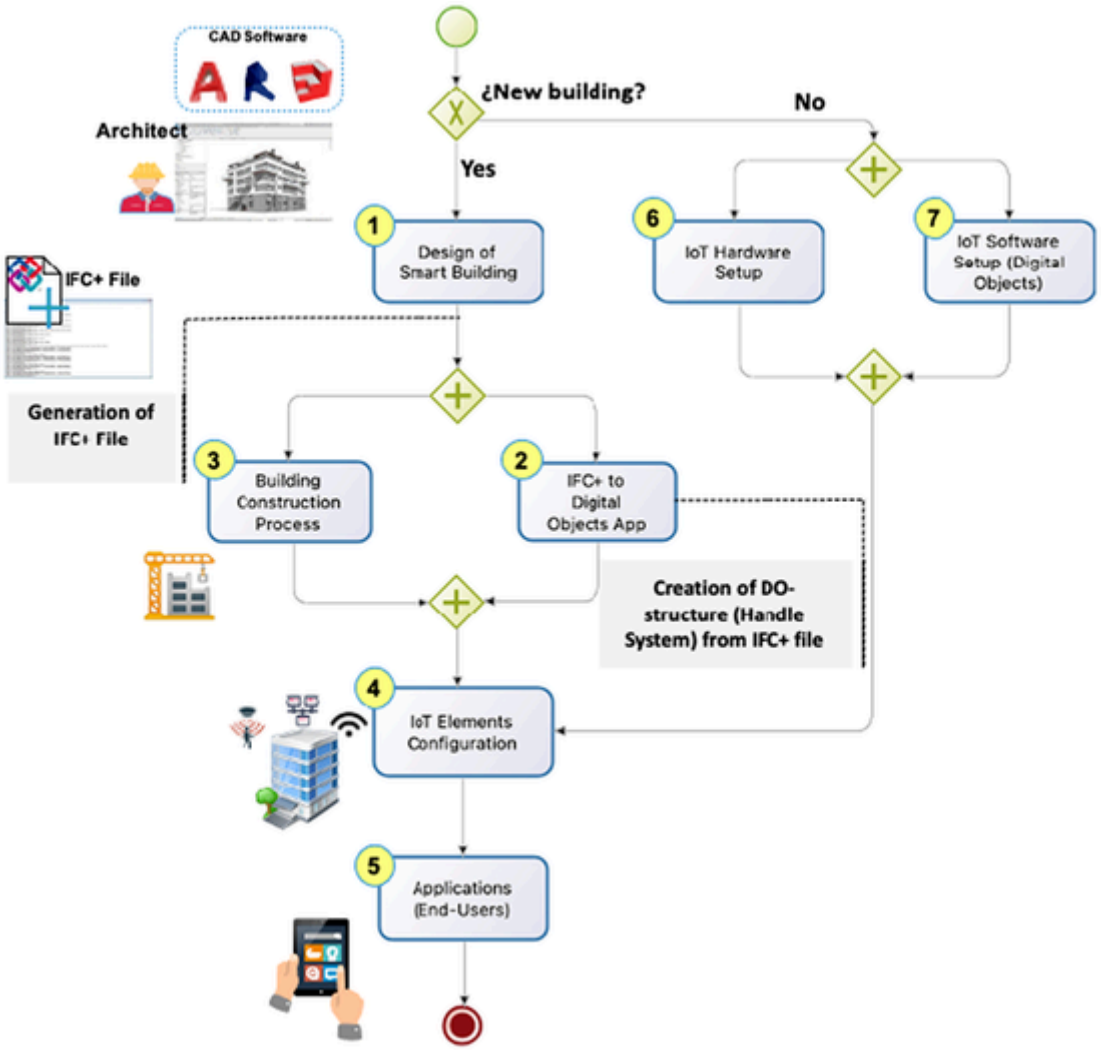


Fig. 2. Approach to support IoT in building design process.

proach is independent and agnostic about the ultimate purpose for which the data produced in a smart building are managed. The stakeholders are free to decide what to do with data once they are collected. This approach also fosters loose coupling between the infrastructure of the building, which may be fully functional aside data, and data processing.

Subsequently, with the IFC + file (which contains a building model), the building construction process is conducted in *Stage 3*. The IoT elements (*i.e.* devices and networks for the building being designed) are configured in *Stage 4*.

Once the building is constructed and IoT elements have been deployed, the end users (*i.e.* maintenance workers, building managers, and office workers) are ready to use smart things (*e.g.* RFID cards, readers, and sensors) through dedicated applications (*e.g.* BAS, BCS, BIS, third-party APIs, and customised applications) in *Stage 5*. These applications interact with the handle system to query features about the IoT elements deployed within the building or, for example, check security constraints before using any device or access to a specific zone. In this stage as well as in the previous stage (Stage 4), IFC + file is not required or used, because the information originally stored in the IFC + file is now implemented and stored in the DO-based infrastructure through the different digital objects, which can be consumed through the high-level REST API. In this way, the applications resolve access permissions to the devices and functionalities available in an IoT scenario (*i.e.* smart building) using the information stored securely in the corresponding DOs.

On the other hand, it may be desirable to add a new IoT-related element to support a new functionality once the building is completed. Although the option to re-start the cycle (from Stage 1 to Stage 5) is always possible, it is much more efficient a new manner to include this IoT element within the constructed building without modifying the CAD model, re-generating the IFC + file and re-constructing the DO infrastructure. In the approach presented, we have defined two stages (Stage 6 and 7) which correspond with the set up of the IoT hardware (installation) and the set up of IoT software required, as it is depicted in Fig. 2. At Stage 7, we can use dedicated software that uses the REST API of the DO infrastructure (Handle System) to create a DO through to represent. Once Stage 6 and 7 are completed, the process can pursuit to the Stage 4, ready to configure the IoT device and use it through the applications or dedicated software (Stage 5).

This approach simplifies the design (construction) and services implementation (software) of smart buildings. The approach enables the design of IoT-based smart buildings by architects, along with the proposal presented based on DOs, allows the automatic generation of ready-to-use web services and consumable by third-party software from the design of a smart building design. That is, the proposal presented enables that the software components (*i.e.* web services), needed to interact and access the IoT devices, can be derived from the specification of such devices in IFC + . In this way, IFC + , together with the approach

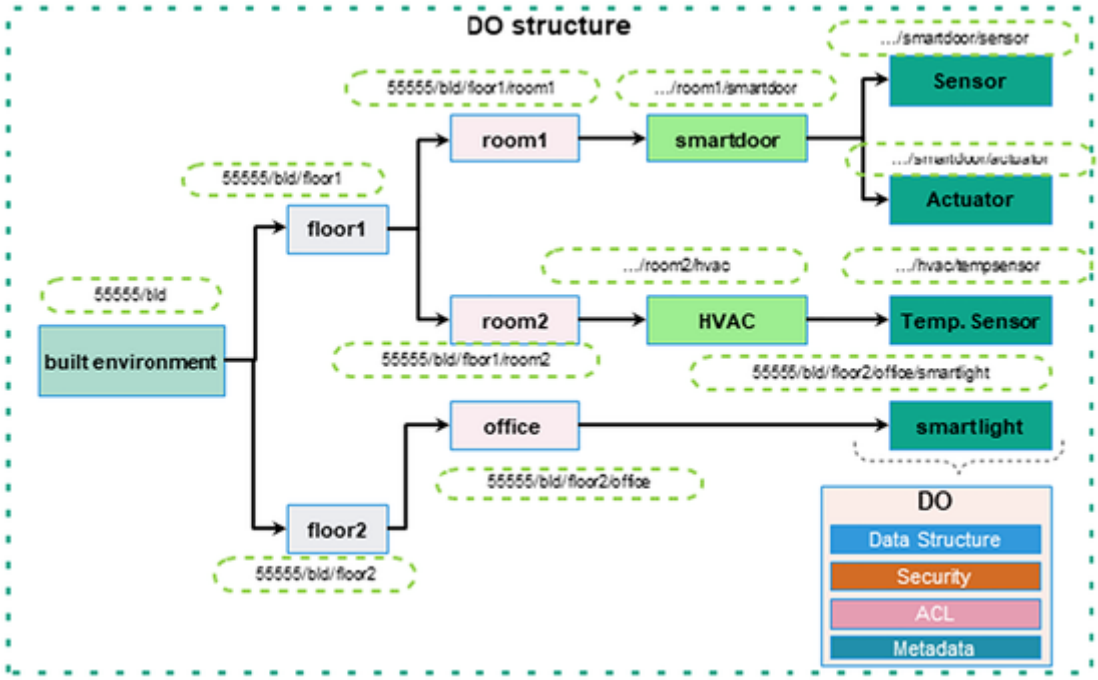


Fig. 3. DO-based hierarchical structure to represent a building through the Handle System.

proposed, work as a bridge between architects and IT guys, because architects without any background about networks or IoT systems, are able to design the building and, IT guys, without knowledge about buildings are able to configure and deploy an IoT infrastructure into a smart building.

4. Proof of concept

In this section, a PoC conducted in a synthetic and simulated built environment is provided to:

- Illustrate the applicability of IFC+.
- Demonstrate the viability of working processes based on the approach presented in Section 3.2 (see Fig. 2), starting from early stages in the construction process of a smart building (*Stage 1*) and concluding with the interaction with IoT devices through mobile apps once the smart building has been constructed (*Stage 5*).
- Evaluate the suitability of IFC+ and the DO-oriented approach in the construction of smart built environments based on BIM.

In the following section, we will present the scenario simulating a fictitious company that plans to use IFC+ and requests the development of a PoC in the laboratory. The remainder of the sections analyse the different stages of the devised process, illustrating the proposed approach (Fig. 2) in a particular and notional scenario.

4.1. Scenario description

A novel and emerging start-up construction company has a project in mind for the design and construction of a smart building, considering the modelling, configuration, and tuning of IoT settings from the early stages of the building construction process, such as device definitions, credentials, and networks. That is, to allow the definition of IoT features during the design stage of a building, rather than after its construction, and transform this design, as automatically as possible, into ready-to-use services of the smart building, which, in turn, will be consumed by end users (*i.e.* building managers, office staff, and visitors) through IoT-based applications or other third-party software (*i.e.* BIS).

The company has its own architectural firm. Hence, according to the current trends and legal requirements, architects design and construct all the buildings using technologies and techniques based on BIM. Because in smart buildings, IoT services (networks, devices, and credential accesses) are set up once the building is constructed, the engineers and architects of the company face two main challenges that hinder their goal of designing a smart building with IoT features:

1. The company uses a proprietary tool (*e.g.* Revit) as BIM software to model the building, and IFC as data representation and exchange format that are used by the different departments and stakeholders. However, either the software or IFC has to be designed to describe sophisticated IoT-based scenarios, such as those deployed in a smart building.
2. Architects and construction engineers lack knowledge about IoT technical problems, which are required for the specification of IoT devices, IoT networks and technology, and configuration parameters required in communication networks.

To overcome these challenges, the company plans to adopt our approach (IFC+ and the DO-based approach) as a possible solution. Before deciding on the procedure to follow to conduct the design and construction of a real smart building, the company requests for a simulation of our solution with an in-the-lab synthetic PoC, which is explained next.

4.2. Application of the proposal

The different stages of the proposed approach (Fig. 2) and the application of the proposed technology are described in detail in the following sections.

4.2.1. Stage 1: building design

The first stage is related to the design of the building, where an architect (or team of architects) uses a CAD software to model the entire building, that is, the different floors, pipes, and electrical installations.

In this PoC, we used a free CAD model available on the Internet[11] and Revit as a CAD software to manage the model (Fig. 4).

In this first stage, and according to the approach presented in Section 3.2, the elements to describe the IoT scenario must be modelled using a CAD software. These elements include IoT devices, networks, users, and credentials, that is, the features supported by IFC+ (Section 3.1).

Because IFC+ is a novel proposal, it is currently not supported by any CAD software. To simulate and illustrate the applicability of the approach, we developed a custom software called *IFC2Handle*.

IFC2Handle is a Java desktop application (see Fig. 5) that provides two main functionalities: *1)* the generation and downloading of IFC+ files from the original IFC files generated by a CAD software and *2)* the automatic construction of the DO-based hierarchical structure in the Handle System using an IFC+ file. In this section, we will focus on the first functionality as the outcome of *Stage 1*. The second functionality will be described later (in *Stage 2*).

IFC2Handle used the file with the IFC model generated by Revit as input (see Fig. 4), which was parsed and displayed in IFC2Handle as a hierarchical view of the building (left-hand side of Fig. 5). From this hierarchical view, it was possible to select those parts of the building that were involved in the IoT scenario, that is, the parts of the building where IoT-related features would be involved, such as devices, credentials, locations, and sensorised spaces.

Furthermore, IFC2Handle is based on automatic model transformations. Transformations and models are defined in terms of IFC models and formal languages, and IFC+ files are instances of the IFC+ metamodel. Likewise, IFC+ files are processed using software constructed over the IFC+ metamodel definition, ensuring the reliability processing.

In this PoC, we considered two rooms of the building as smart rooms (*i.e.*, sensorised rooms). They appear highlighted in blue in Fig. 4 and identified as *R1* and *R2* in the left-hand side of Fig. 5.

Once certain zones of the building were selected as smart spaces, we generated the corresponding IFC+ file by clicking on the *'Generate IFC+ File' option*. This IFC+ file could be downloaded through the application, when required, using the *'Download IFC+' button*. Fig. 6 displays an excerpt from the IFC+ file generated in this case. The entire file is available in the link provided in the'Supplementary Material' section.

4.2.2. Stage 2: transformation of IFC+ entities into DOs

As mentioned before, the IFC2Handle application implements the Stage 2 of the approach, that is, the transformation of IFC+ entities into DOs which can be managed through the Handle System (see Fig. 2). The different DOs were created using a hierarchical structure, preserving the specification of the structure of the building described in the IFC file. The IFC2Handle application used a custom parser to conduct this task, which was also implemented to transform the IFC+ files into programming language objects (*i.e.* Java programming language objects). The parser, called *IFCPlusParser*, is available for developers/programmers as a Java library in the repository referenced in the "Supplementary Materials" section, which implies that it can be imported in any Java-based project.

On the right-hand side of the graphical user interface (GUI) of the application (see *'Project Options'* in Fig. 5), the parameters required to set up the DO-based hierarchical structure in the Handle System must be defined, that is, the name of the project, prefix, credentials to access the Handle System (Handle Admin, and Password), and the Project Manager (PM) credentials (PM Username and Password). The PM, in this PoC, was the user in charge of the building (*e.g.* a building manager).

Consequently, a local Handle System was obtained, deployed with the basic DO structure that represented the IoT-related parts of the building. In this case, two different DOs were created to support the two smart rooms: *55555/maletplace/engbuilding/level3/space/r1* and *55,555 /maletplace/engbuilding/level3/space/r2*.

IFC2Handle application is available in the link provided in the'Supplementary Material' section.

4.2.3. Stage 3: building construction process

This stage encompassed the construction process of the building using BIM methods and techniques, concluding with the building constructed after passing through the construction tasks required (*e.g.* obtaining planning permissions, pouring the foundation, framing, and electrical and plumbing installations). This stage included several different tasks involving several experts (*e.g.* architects, plumbers, and structure engineers) and processes in several areas. Furthermore, for a smart building, IoT-related elements must be installed in this stage (*e.g.* network installation, sensors and actuators, and smart doors), similar to other installations such as pipes or electrical wiring.

4.2.4. Stage 4: deployment of the IoT scenario

Once the building was constructed (Stage 3) and the digital representation of the IoT-related elements of the building were represented as DOs in the Handle System, the IoT elements (devices and networks) were installed and configured. Ideally, the installation of IoT elements should be conducted during Stage 3, as part of the building construction, similar to the electrical or plumbing infrastructure. For simplicity and pragmatism, to illustrate the simulation, which is not a complete BIM process, everything related to the IoT (installation and configuration) is addressed at this stage.

Two main tasks related to IoT were conducted at this stage: *1)* the installation of IoT elements and *2)* the configuration and implementation of the software to interact with the IoT elements to provide the services required in the smart building. With these premises, several concerns were identified to make the PoC as similar as possible (in terms of technology used and behaviour expected) to a real deployment of a smart building. They are:

- A network is an important infrastructure asset in any IoT scenario because it enables the communication between different devices and applications and provides services to the end users of the smart building. Therefore, it is necessary to select an IoT-compliant network that is suitable for smart building environments.
- A smart building communicates with devices (sensors) to obtain information from the environment and also to interact with devices (actuators). To illustrate how to address this type of situation within the running simulation example, we assumed that the following elements were located in room 1 (R1): a smart light, a temperature sensor, a humidity sensor, and an air conditioning machine (actuator). In this PoC, we used commercial and low-cost consumer electronic products to simulate the real elements.
- Real-time operation is vital in any smart building environment to provide smart services (*i.e.* management of the building and access to environmental features). Therefore, we must use a proper technology that guarantees access to information in real-time.

These concerns are addressed in the following sections.

4.2.4.1. IoT network. Several network technologies are available to support communications within a smart environment, most of which are wireless networks. Among them, LoRaWAN is currently the most used network for both indoor and outdoor spaces. LoRaWAN is an LP-WAN that provides several benefits that are highly suitable for IoT sce-

[11] https://grabcad.com/

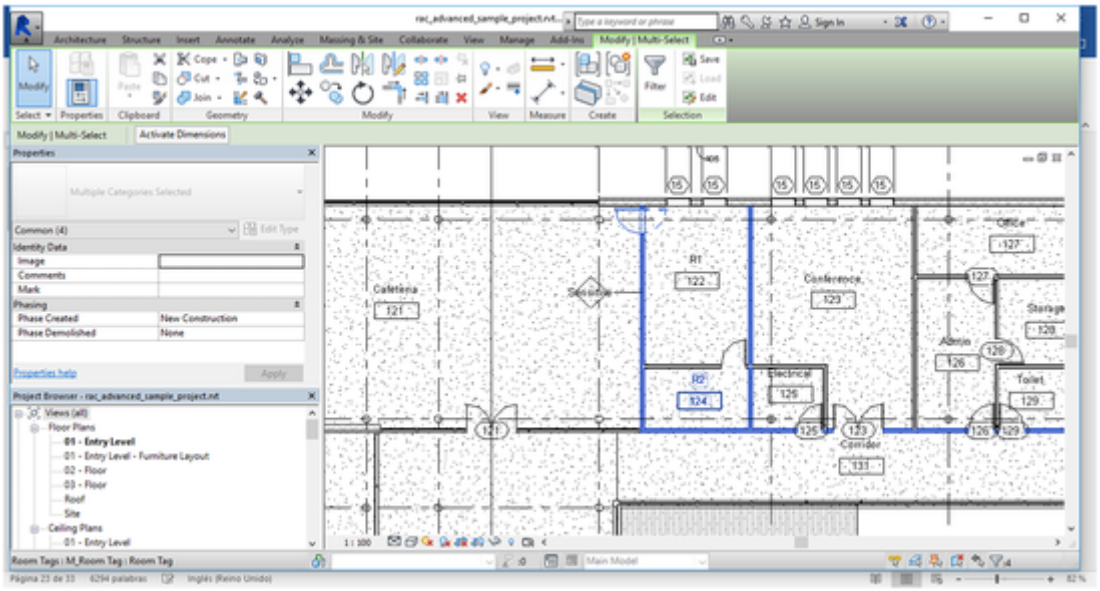


Fig. 4. Building CAD model.

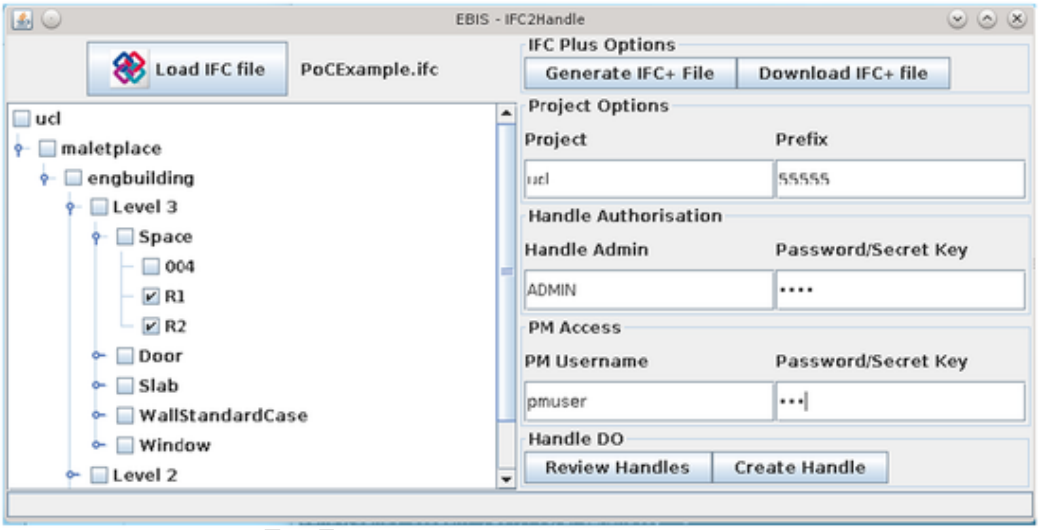


Fig. 5. IFC2Handle: Software to manage IFC+ file.

narios, such as low power consumption, low-cost connectivity, low cost of LoRaWAN-based devices, and long-range capability [49]. In reality, LoRaWAN has been widely used in several projects and studies related to smart environments as IoT networks [56–58] and hence, is the one selected for this PoC.

LoRaWAN follows a deployment model based on four main elements (Fig. 7):

1. *End Nodes*. LoRa embedded devices, which typically have sensors/actuators to interact with the environment, a LoRa transponder to transmit signals (data), and optionally, a microcontroller.
2. *Gateways*. LoRa gateways act as bridges between end nodes (*e.g.*, LoRa transponders) and network servers, which use the IP.
3. *Network Servers*. It connects gateways with *Application Servers* (see below), working as a router, broker and handler, processing all data packets from the LoRaWAN gateways.
4. *Application Servers*. They store the information gathered by end nodes, allowing their consumption by end users through any software or application.

The EveryNet platform,[12] which automatically allows the use of public LoRaWAN infrastructure, was used as the network server in this PoC. The EveryNet platform automatically detects nearby LoRaWAN public gateways and uses a private network server to gather information from the end-node devices and dumps the data into an application server. Furthermore, EveryNet enables the management of end-node devices and LoRaWAN secure-related parameters (*i.e.* Network Session Key, Application Session Key, Device Address, and Device Unique Identifier) [49] through a friendly front-end, simplifying the setup of LoRaWAN features and configuration.

The PubNub platform[13] was used as an Application Server in this PoC which enabled data storage and provided a high-level REST API to developers/consumers with access to these data from any device, anywhere and anytime, allowing the consumption of such data by any other application or customised software.

The end-node devices used for this PoC are extensively detailed in the following section.

[12] https://www.everynet.com/
[13] https://www.pubnub.com/

```
#48048= IFCCARTESIANPOINT((0.0,0.0,0.0));
#48050= IFCBOUNDINGBOX(#48048,5200.0,1200.0,2400.0);
#48051= IFCSHAPEREPRESENTATION(#157,'Box','BoundingBox',(#48050));
#48053= IFCPRODUCTDEFINITIONSHAPE($,$,(#48045,#48051));
#48057= IFCSPACE('3sstO$18JoIeuAwWsqJXGx',#15,'R2',$,$,#48020,#48053,'Corridor',.ELEMENT.,.INTERNAL.,$);
#48061= IFCPROPERTYSINGLEVALUE('Category',$,IFCLABEL('Exit lobby'),$);
#48062= IFCPROPERTYSET('3q3vo6TJHUPmmmB7qMSWzb',#15,'Pset_SpaceCommon',$,(#48061));
#48064= IFCRELDEFINESBYPROPERTIES('1cWumUvl39oPkAdDM6ppbb',#15,$,$,(#48057),#48062);
#48068= IFCPROPERTYSINGLEVALUE('Full Element ID',$,IFCLABEL('Zone - 017'),$);
```

(a) R2 as *IfcSpace* entity of the original IFC description (id: 48057)

```
#55638= IFCCARTESIANPOINT((0.0,2400.0,0.0));
#55640= IFCCARTESIANPOINT((5200.0,2400.0,0.0));
#55642= IFCCARTESIANPOINT((5200.0,0.0,0.0));
#55644= IFCCARTESIANPOINT((0.0,0.0,0.0));
#55646= IFCCARTESIANPOINT((0.0,2400.0,0.0));
#55648= IFCPOLYLINE((#55638,#55640,#55642,#55644,#55646));
#55650= IFCCOMPOSITECURVESEGMENT(.CONTINUOUS.,.F.,#55648);
#55651= IFCCOMPOSITECURVE((#55650),.F.);
#55654= IFCCURVEBOUNDEDPLANE(#55637,#55651,());
#55656= IFCCONNECTIONSURFACEGEOMETRY(#55654,$);
#55657= IFCRELSPACEBOUNDARY('30Qf3spcDrF4W7EozfAWvh',#15,'2ndLevel','2a',#48842,#43394,#55656,.PHYSICAL.,.EXTERNAL.);
#55658= IFCPLUSIOT($,$,$,$,$,'449dc7dc-b5a9-4058-b963-3cded519d4f6');
#55659= IFCPLUSRELIFC4($,#48057,#55658);
#55660= IFCPLUSIOT($,$,$,$,$,'24e70fb3-2a9a-4a23-a019-aacd3a796063');
#55661= IFCPLUSRELIFC4($,#48842,#55660);
ENDSEC;

END-ISO-10303-21;
```

(b) Transformation of R2 room into smart room through *IfcPlusRelIFC4* entity

Fig. 6. IFC+ file excerpt.

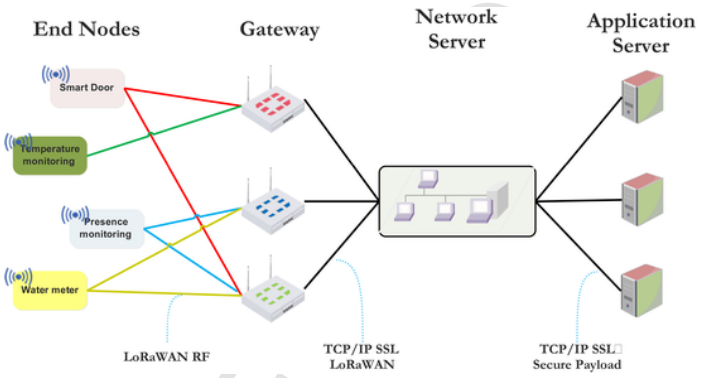


Fig. 7. LoRaWAN deployment model.

4.2.4.2. IoT devices (LoRaWAN end node). As mentioned before, one of the main concerns in any smart environment is the interaction with the environment through devices (sensors/actuators).

In this PoC, we considered only R1, where one smart light, a temperature/humidity sensor, and an air conditioning machine were located. For this simulation, low-cost electronic products were used, that is, a red LED for the smart light, a commercial temperature/humidity sensor model DHT11,[14] and a motor with a fan blade to act as the air conditioning machine.

As an end-node LoRaWAN device, we used Lopy4 technology, a compact quadruple-network MicroPython-enabled development board where the different electronic products (red led, temperature/humidity sensor, and fan) were connected.[15] The end-node (LoPy4) enabled the connection and gathering of information from sensors/actuators and implemented the LoRaWAN RF protocol, allowing the communication between the end-node and the gateway, and therefore, the data exchange and interaction between the environment and end users through the LoRaWAN infrastructure elements (*i.e.* Gateways, Network Servers, and Application Servers).

4.2.4.3. Real-time environment interaction. Support for real-time features is essential for most of the services provided by a smart environment, such as obtaining information in real-time (*i.e.* temperature or if a light is on or off) as well as sending order and acting immediately on the environment (*i.e.* turn on a light or set the temperature of the room through the air conditioning machine).

To support real-time reactions in this IoT scenario, several IoT-oriented technologies exist, such as the so-called Machine to Machine (M2M) technologies. These technologies enable the interaction and exchange of information between software and devices without human intervention, for instance, between a temperature sensor (or end-node LoRaWAN device) and a mobile application. For this PoC, we chose Message Queuing Telemetry Transport (MQTT) as M2M technology [59]. MQTT is a communication protocol supported by a flexible architecture that implements a publisher-subscriber design pattern, where subscribers (end-point elements, that is, software, application, and artefacts) can subscribe to a specific publisher or topic (*e.g.* the last temperature measured in a room or the state-open, closed-of a door). Once certain information is sent by the publisher (*e.g.* updating the temperature of the room or the notification that a door has been closed), it is automatically received by the corresponding subscribers through the MQTT broker (a server that receives all messages from the publisher and then routes the messages to the appropriate subscribers).

Several commercial and open-source solutions are available to implement MQTT (*e.g.* Google IoT Core, Microsoft IoT Azure, and Eclipse Mosquitto). The PubNub platform, used as Application Server in the LoRaWAN infrastructure for this PoC, provided an MQTT service that is easily interoperable with any IoT application. Thus, the PubNub platform enabled the automatic forwarding of information gathered from end-node LoRaWAN devices through the use of the gateway into to the

[14] https://components101.com/dht11-temperature-sensor
[15] https://pycom.io/product/lopy4/

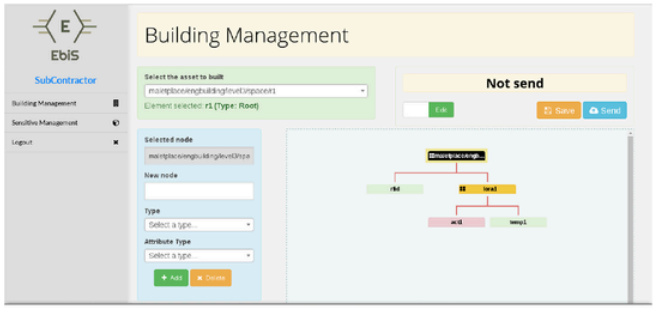
Network Server; which then forwarded the information to the PubNub-supported Application Server.

4.2.5. Stage 5: use of applications for the management of the building and DOs

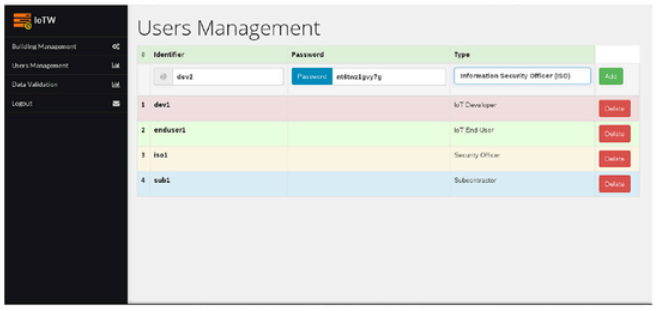
In this PoC, we established *R1* as the room with IoT devices in the building. Although the associated IFC+ schema is available and its generation has been described in Stage 1, we require additional software to manage the DOs that represent the IoT settings of the smart building (*i.e.* devices located in R1, users and credentials, sensors, and actuators).

We developed a software platform, called *IoTW*, that allowed architects and developers to manage the DOs of a building. This feature improved the management of the elements related to the IoT scenario of the building, from the creation of new DOs to the representation of new IoT devices for the management of credentials, where different users of the building were also represented as DOs.

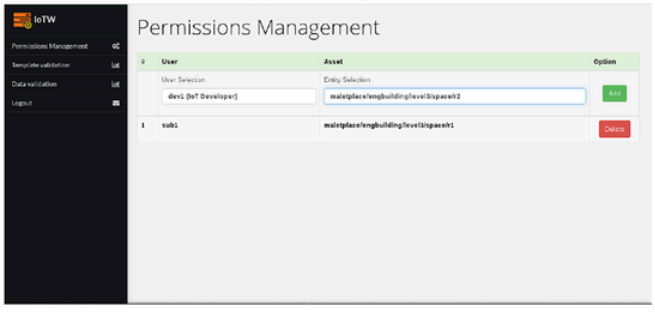
IoTW is a multi-stakeholder platform where each authorised type of user (*e.g.* building manager and building staff) has a custom GUI that provides a specific set of functionalities, for instance, create new users, set permissions, add new assets to the building, and design data structure for each asset (*i.e.* templates [55]). IoTW was implemented in NodeJS for the back-end and in AngularJS and pure JavaScript for the front-end. Fig. 8 depicts the main screenshots of *IoTW*.



(a) Addition of new IoT elements to the Room1 (R1)



(b) Users management



(c) Access control system of the building

Fig. 8. IoTW web platform for the management of Digital Objects through the Handle System.

Additional software is required to monitor and interact with buildings, such as BIS or custom applications, which must also be used in this stage. We implemented a lightweight Android-based application to interact with the temperature sensor located in R1. In this application, the user (*e.g.* a maintenance operator created in the IoTW platform earlier –see Fig. 8b), could log into the application to obtain the temperature value of (*R1*).

The application would request the handle system for access to the DO that represented the temperature sensor. If the user was granted permission, he/she could obtain the required information to access the sensor information. In this case, the user obtained the subscriber token required to gather the information broadcast by the MQTT server (supported by PubNub). The temperature was received in the mobile application and displayed on the screen, as illustrated in Fig. 9.

4.2.6. Stage 6 and 7: IoT Setup after building construction

New IoT elements can be added some time later if required, once the building is constructed.

Once the IoT elements are installed, physically, in the building (Stage 6), the software developers or IT guys are able to represent these new elements as DOs, and then, configure them (Stage 4) and using them through the proper application (Stage 5). This could be done through a third-party application that uses the REST API provided by the DO-infrastructure, such as the *IoTW* web platform, which already displays in 4 an interface to add new IoT elements to a part of a building (R1) once the building is already constructed.

4.3. PoC design and workflow

In addition to the construction processes related to BIM, a proper design of the IoT scenario is required. To design the IoT scenario that we intend to simulate in this PoC, we used an architectural model to describe the different elements and their relationship [60]. A common IoT architecture is a layered oriented architectural model that includes three different layers: *1) the Perception Layer (also called Acquisition Layer), 2) Network Layer, and 3) Application Layer* [61].

In this architecture, all sensing devices (IoT devices) are placed in the Acquisition layer, which can be used by IoT applications (Application Layer) through the Network Layer (gateways, network servers). A detailed explanation of this architecture can be found in [22].

The PoC presented was designed according to this reference architecture. Fig. 10 depicts the workflow followed to develop the PoC de-

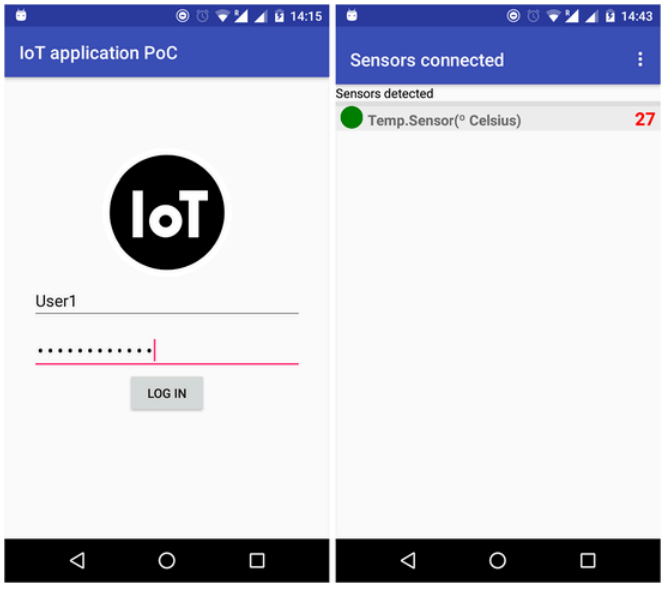


Fig. 9. Application to interact with the building.

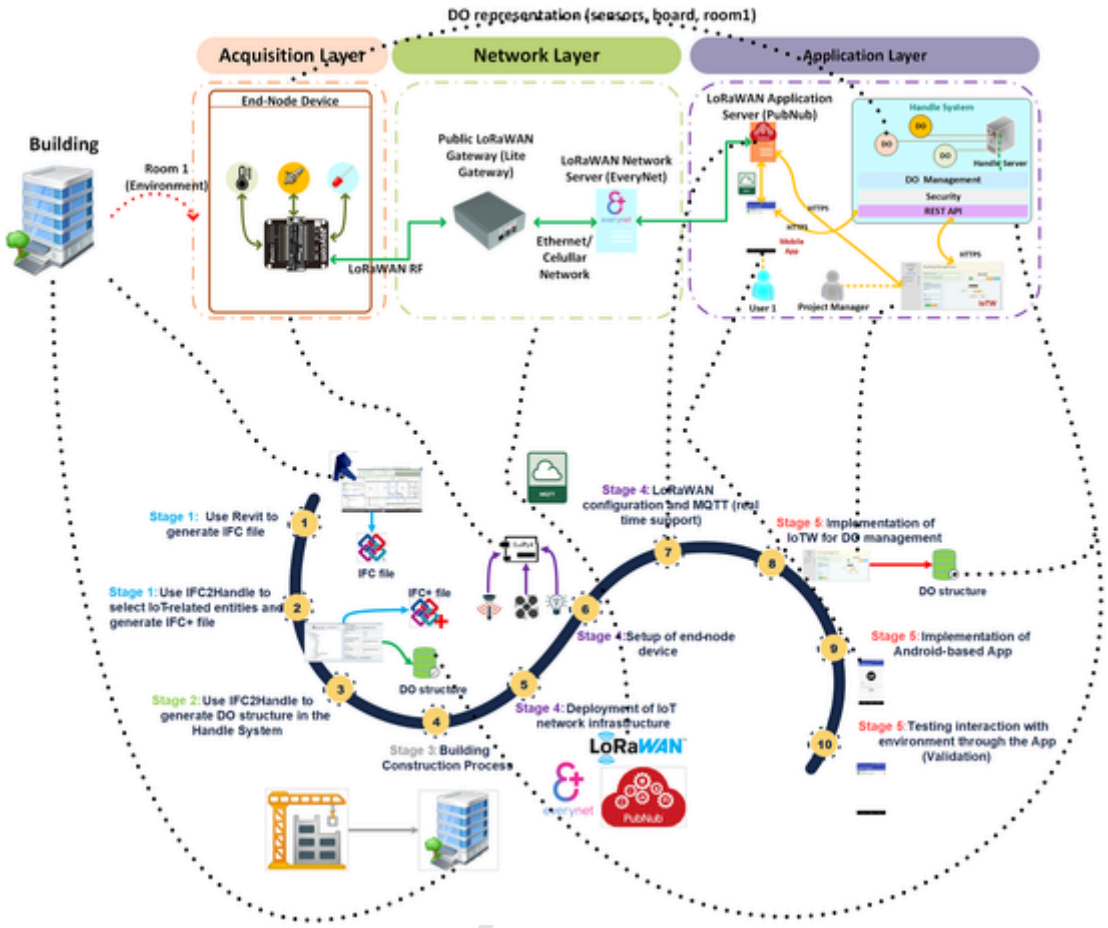


Fig. 10. PoC workflow and architecture.

scribed in this section, which matches the proposed approach (Section 3 and shown in Fig. 2). In addition, (Fig. 2) illustrates the relationship between the workflow of the PoC and the three-layer oriented architecture used to design and describe the IoT scenarios.

4.4. Suitability and evaluation of the contributions

Through the PoC we demonstrated how IFC+ and the DO-based approach could be used to *1)* support the design of IoT scenarios since the early stages of building design and *2)* allow the representation of these scenarios through consumable software (DOs) directly by IoT applications and end users.

Meanwhile the adoption and support for IFC+ by mainstream CAD vendors have become a reality we have implemented certain simple applications to illustrate the feasibility of planning building design processes that include the representation of IoT elements in the early stages.

Ideally, once CAD software used in BIM for constructions (*i.e.* Revit and AutoCAD) implements the IFC+ schema specification, it could be possible to design, similar to other assets, furniture, or installation, IoT scenarios, specifying the different IoT devices, networks, credentials, and safe zones with restricted access. Once the entire building was designed, including IoT-related elements, the architect could, through the same CAD software, generate the IFC+ file to share it with other experts or architects and automatically deploy the entire DO-based infrastructure supported by the Handle System in a transparent manner and without any specific technological background (*i.e. seamlessly*).

IoT applications could consume the different DOs that represent the IoT-related elements in the smart environment through the REST API provided by the Handle System. The use of this native REST API reduced the development time of IoT applications, because developers did not have to dedicate time and effort in designing and implementing the back-end services related to the representation and storage of information for the smart environment, which were automatically managed by the Handle System.

In this PoC, we simulated the approach described in Fig. 2 by implementing custom software (*i.e.* IFC2Handle, IoT applications and IFCPlusParser) to address and overcome the technological challenges that exist in the implementation of the proposed approach.

5. Discussion

The solution presented in this study was designed to enable the modelling of IoT scenarios along with the design of the building. The different contributions have outstanding benefits:

- The use of IFC+, as an extended schema of the current version of IFC (version 4), enables the description of IoT scenarios from early stages in the design of buildings in BIM processes.
- The approach presented simplifies the development of IoT solutions because IoT-related features described in IFC+ are automatically translated into software artefacts (DOs) supported by the Handle System.
- DOs, as digital representations of IoT-related elements involved in the building, could be consumed by any type of application through a REST API, which fosters the agile implementation of IoT applications and the integration with BAS, BCS and BIS.
- The use of secure DOs supported by the Handle System enhances the security concerns of smart spaces, providing for instance, ACLs

for each IoT-related device or element, and secure access to sensitive data together with the management of credentials.
- A hierarchical structure to digitally represent a smart space simplifies the management of the supplies and the credentials/authorisation through a permission inheritance mechanism.

Conversely, the proposal presented still has some challenges and limitations.

IFC+ is a novel proposal; hence, no platform or CAD software integrates the IFC+ schema (and its features) yet. This implies the need of additional software to modify the original IFC model to add IoT features and transform it into an IFC+ model, such as *IFC2Handle* software, the example described in Section 4.

IFC+ has been designed and created as an extension of IFC (v4) schema. Automated software tools (in the form of parsers and other processing software) are also involved in producing and processing the extended metamodel of IFC+. The core entity of IFC+, *IFCPlusIoT*, is directly a sub-entity of the *IfcObject* entity, originally defined in the IFC schema. In IFC+ all the entities inherit, directly or indirectly, from *IfcPlusIoT* entity. So, all the IFC+ entities are sub-entities, at different hierarchical levels, of *IfcObject*. Thus, IFC+ inherits all benefits (and limitations) of IFC concerning the representation of IFC scenarios using IFC+. This entails that, as for the reliability of IFC scenarios, IFC+ preserves the same reliability as IFC. Likewise, IFC provides mechanisms to link the different construction elements through several properties of the IFC metamodel, and the same applies to the IFC+ extension, so that IFC+ IoT devices can be also linked to traditional construction elements present in the IFC metamodel and other IFC+ elements.

However, while for IFC, some CAD software (like Revit) provide enhanced reliability of models through different conformance checks, the same support is not currently available for IFC+ as it is in the early stages of its development. For instance, Revit implements automatic background processes to, using the position of the elements and the type, check if the position and the assembling of these elements are correct or not, displaying a warning message in case something is not ok. The *IFC2Handle* tool has been developed as a proof-of-concept software about the feasibility of the presented IFC+ metamodel extension. Further work is needed to evolve this and other tools that carry out additional consistency and conformance checks to improve the reliability of IFC+ scenarios.

In addition, even if IFC+ was supported by a CAD software (AutoCAD, Revit, and Sketchup), architects typically do not have the technological background required to model IoT scenarios and set up network configurations or IoT-based devices. Thus, CAD software that integrates IFC+ should provide a friendly GUI to assist architects in modelling IoT scenarios or facilitate the involvement of users with the technological background that is required in the modelling process of a building in BIM.

Furthermore, another possible approach could be the definition of multi-stakeholder processes in BIM, where different architects and technicians in computing coexist in the early stages of building design. Through the traditional CAD software, architects model the built environment and generate the corresponding IFC file as output. This IFC file could be used as input by technicians in an additional software created to complete the model of the building with IoT features and generate the IFC+ file. This approach, addressed in the PoC through *IFC2Handle software* (see Section 4), is the most suitable approach to the widespread use of the IFC+ schema, because no changes are required in traditional BIM modelling processes.

The current state of IFC+ is the outcome of a custom design to support most of the relevant elements related to IoT in smart spaces, but not all of them are supported in the current version. Although traditional supplies and furniture described in IFC do not noticeably change over time, IoT and smart spaces are constantly evolving, owing to the emergence of novel technologies and requirements. Therefore, IFC+ must be updated from time to time with new entities, relationships, and attributes, adapting over time to trends in smart spaces. This is not a concern because the IFC+ schema can be published in a public repository, and distributed knowledge graph technologies can help in keeping track of the versions of the schema and compatibility issues [62].

Finally, the definition of IFC+ as an extension of the original IFC schema has several other implications related to its use, design, development, and management of periodical updates.

First, programming skills would be required to improve or extend the IFC+ in a functional and operable way, which is not accessible for an average architect, designer, or construction-related engineer.

Second, and finally, according to buildingSMART (formerly, the International Alliance for Interoperability -IAI-), the creator of IFC, there are three different ways to extend IFC: *1)* defining new entities or types, *2)* using proxy elements, and *3)* using the property sets or types [63]. The first option (the definition of new entities/types) was the one chosen to design and create IFC+ by adding entities (*IFCPlusXXX*). This is the best approach to extend the original IFC schema because the new entities and types can be used in the same manner as the existing ones, whereas with the other two options, additional implementations are required [64]. However, buildingSMART takes two or more years to integrate the proposed extensions in a new IFC release [63]. Therefore, the time frame required for the outcomes of this study to pass through the certification phase of buildingSMART to improve the current IFC schema with IoT-related features presented in IFC+ is too long for a dynamic domain such as IoT; hence, the subsequent versions of IFC that include IFC+ entities/features may be outdated when released.

However, it is important to mention that further development of IFC+ is required to ensure a ready-to-use solution and an IFC+ schema ready to be considered as a possible extension of the current IFC schema by buildingSMART. We are convinced that a solution such as IFC+ is essential in the near future to support BIM-based constructions with IoT elements as key concerns for any smart environment.

6. Conclusions

The integration of IoT in BIM has the potential to transform the way of interacting with buildings, where IoT-based technology provides an opportunity to enhance user experience. Currently, a growing concern exists about the integration of IoT into BIM. Most existing solutions and approaches are oriented to integrate IoT applications through BIS and BCS, once the building is constructed. However, the integration of IoT from the early stages of BIM, that is, during building design, is not a commonly addressed topic, although it can provide important benefits, such as the enhancement of the design of smart environments, the possibility of adapting IoT scenarios to new requirements through modelling mechanisms, and the use of technologies to automatically deploy ready-to-use IoT solutions from models.

This study provides an approach to integrate IoT into building models in early stages of design in BIM processes, through two main contributions: *1)* IFC+, an extension to IFC that enables the representation of IoT devices and related elements, and *2)* a DO-based approach to transform building models with IoT features into working software components to support IoT interaction scenarios. In particular, IFC+, an extended schema of the latest version of the IFC (version 4), supports the modelling of IoT-related features with security constraints (Things, Locations, Networks, State, Security). The proposed approach enhances the integration of IoT and BIM by providing the mechanisms and tools required to transform, automatically, IoT-related elements described in IFC+ into secure DOs, supported by the Handle System and accessible by external software through a friendly REST API.This automatic translation of IFC+ entities and DOs reduces, significantly, development times, integration efforts with third-party software and reduces config-

uration and specification errors in the API caused by human intervention, as several pieces of software are automatically generated.

As an outcome of this study, we illustrated the feasibility of this approach with an in-lab use case simulation. Other use cases simulation can be tested and validated using the software provided in the *Supplementary Section*. The *IFC2Handle* is used to transform a CAD exported IFC file into an IFC+ file which automatically transforms the IFC+ file into ready-to-use DOs.[16] The Java-coded parser provided could be used in case it is desired to integrate the support or compatibility of IFC+ into custom applications.

As future work, we envision to improve IFC+ by adding new entities to support cutting-edge IoT technologies and cover other IoT features unavailable in the current version for smart buildings, such as other authentication mechanisms, additional types of networks with different topologies, data exchange formats present in IoT devices (*e.g.* Generic Attribute Profile -GATT- specification in Bluetooth), and support available standards for the IoT world. Furthermore, it is desirable to work on the dissemination and broadcast of IFC+ and related approaches, such that CAD software vendors and related stakeholders start to integrate these extensions in their products and validate their suitability with architects and other practitioners from the built domain.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This paper was funded by the Spanish Ministerio de Ciencia, Innovación y Universidades under contracts RTI2018-100754-B-I00 (iSUN project) and RTI2018-098160-B-I00 (Air Forecasting project). In addition, this work also received inputs from the COST Action CA19134 Distributed Knowledge Graphs.

[16] Previous installation of the Handle System is required. Please check https://hdl.handle.net/20.1000/113

Appendix A. Entities added in IFC+

Table A.1
IFC+ entities description.

Group	Entity	Description
Root	IfcPlusIoT	Main entity to represent a Thing and linked to IFC (subclass of IfcObject)
Things and Networks	IfcPlusNetwork	Entity to describe an IoT-based network
	IfcPlusTag	Entity to represent a Tag, that is, a device to be read by a reader
	IfcPlusReader	Entity to represent a badge reader device
	IfcPlusGateway	Entity to represent a gateway into an IoT-based network
	IfcPlusDevice	Entity to represent a physical device (sensor, actuator, tag, HVAC)
	IfcPlusGroupIoT	Entity used to group related entities according to specific features (location)
	IfcPlusLoraDevice	Entity to represent an end-node LoRa device (mote)
	IfcPlusSensorDataField	Entity to represent relevant information for each value provided by a sensor
	IfcPlusSensorDescriptor	Entity used to represent the descriptor of a sensor
	IfcPlusSensorMetadata	Entity for the metadata associated with a specific sensor
	IfcPlusServer	Entity to represent a physical server into an IoT scenario
State	IfcPlusState	Entity defined as super-object to represent the state of a Thing
	IfcPlusStateDoor	Entity to represent the state of a door
	IfcPlusStateWindow	Entity to represent the state of a window
	IfcPlusStateLamp	Entity to represent the state of a lamp
Location	IfcPlusLocation	Alternative way to specify the location of a Thing through an array of elements
Security	IfcPlusCredential	Entity to represent the authorised users for a specific Thing
	IfcPlusSecurity	Entity to represent method used in the security of a Thing
	IfcPlusLoRaSecurityNode	Entity for the secure-related parameters of an end-node LoRa device

Group	Entity	Description
	IfcPlusLoRaSecurityServer	Entity for the secure-related parameters of a LoRa server
Agent	IfcPlusAgent	Entity to represent an agent (software, device) that could interact with a Thing
	IfcPlusRole	Entity to describe the role of an Agent
Relationships	IfcPlusRelIFC	This relationship is used to link IFC+ entities with IFC entities
	IfcPlusRelDeviceLora	Relationship to link an end-node LoRa device as a Thing
	IfcPlusRelLocationIoT	Relationship to link a location with a Thing
	IfcPlusRelGroupIoT	Relationship to link a Thing with a group of Things
	IfcPlusRelNetworkDevice	Relationship to link a Thing with a network
	IfcPlusRelSecurityDeviceLora	Relationship to link a LoRa secure specification with a Device/Thing
	IfcPlusRelSensorData	Relationship to link a data specification with a sensor
	IfcPlusRelSensorInterface	Relationship to relate a sensor with a specific interface description
	IfcPlusRelSensorMetadata	Relationship to link a metadata with a specific sensor
	IfcPlusRelServerSecurity	Relationship to link a security specification with a server
	IfcPlusRelDataFieldState	Relationship to related a data field with an state of a Thing
	IfcPlusRelAgentRole	Relationship to link a specific role of an agent with a Thing
	IfcPlusRelSecurityIoT	Relationship to link a security specification with a Thing
Types	IfcPlusState	OPEN, CLOSE, HALF_OPEN, ON, OFF, UP, DOWN
	IfcPlusStateLevel	Represent the state level of a Thing as a percentage, fromt 0 to 100 (percentage)
	IfcPlusNetworkTopology	POINT_TO_POINT, RING, STAR, MESH, BUS, TREE, HYBRID
	IfcPlusNetworkProtocol	WIFI, ETHERNET, BLUETOOTH
	IfcPlusNetworkType	CLIENT_SERVER, P2P
	IfcPlusTagState	READY, ARBITRATE. REPLY, ACKNOWLEDGED, OPEN, SECURED, KILLED
	IfcPlusReaderInterface	I2C, SPI, SMBus
	IfcPlusReaderState	READ, WAITING, READING, BLOCK
	IfcPlusSensorDescriptorSeparator	DECIMAL, TOKEN, BLOCK
	IfcPlusSensorUnitMeasure	Represent the different unit measures, according to Sensor Markup Language (IETF)
	IfcPlusSecurityEncryptionMethod	DES, RSA, SHA, AES
	IfcPlusSecurityAuthenticationMethod	HASH, MAC, DIGITAL_SIGNATURE
	IfcPlusAgentRole	OWNER, AUTHORISED, NOT_AUTHORISED, AUTHORISED_TEMPORARILY

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.autcon.2022.104129.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE Commun. Surveys Tutorials 17 (4) (2015) 2347–2376, https://doi.org/10.1109/COMST.2015.2444095.

[2] S. Li, L. Da Xu, S. Zhao, The internet of things: a survey, Inf. Syst. Front. 17 (2) (2015) 243–259, https://doi.org/10.1016/j.comnet.2010.05.010.

[3] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, A survey on internet of things: architecture, enabling technologies, security and privacy, and applications, IEEE Internet Things J. 4 (5) (2017) 1125–1142, https://doi.org/10.1109/JIOT.2017.2683200.

[4] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): a vision, architectural elements, and future directions, Futur. Gener. Comput. Syst. 29 (7) (2013) 1645–1660, https://doi.org/10.1109/I-SMAC.2017.8058399.

[5] A. Ruiz-Zafra, K. Benghazi, C. Mavromoustakis, M. Noguera, An iot-aware architectural model for smart habitats, in, in: 16th IEEE International Conference on Embedded and Ubiquitous Computing (EUC), IEEE, 2018, pp. 103–110, https://doi.org/10.1109/EUC.2018.00022.

[6] A. Khanna, S. Kaur, Evolution of internet of things (iot) and its significant impact in the field of precision agriculture, Comput. Electron. Agric. 157 (2019) 218–231, https://doi.org/10.1016/j.compag.2018.12.039.

[7] D. Minoli, K. Sohraby, B. Occhiogrosso, Iot considerations, requirements, and architectures for smart buildings—energy optimization and next-generation building management systems, IEEE Internet Things J. 4 (1) (2017) 269–283, https://doi.org/10.1109/JIOT.2017.2647881.

[8] B. Morvaj, L. Lugaric, S. Krajcar, Demonstrating smart buildings and smart grid features in a smart energy city, in: Proceedings of the 3rd International Youth Conference on Energetics (IYCE), IEEE, 2011, pp. 1–8.

[9] G.T. Costanzo, G. Zhu, M.F. Anjos, G. Savard, A system architecture for autonomous demand side load management in smart buildings, IEEE Trans. Smart Grid 3 (4) (2012) 2157–2165, https://doi.org/10.1109/TSG.2012.2217358.

[10] S. Azhar, M. Khalfan, T. Maqsood, Building information modelling (bim): now and beyond, Construct. Econ. Build. 12 (4) (2012) 15–28, https://doi.org/10.5130/ajceb.v12i4.3032.

[11] R. Howard, B.-C. Björk, Building information modelling–experts' views on standardisation and industry deployment, Adv. Eng. Inform. 22 (2) (2008) 271–280, https://doi.org/10.1016/j.aei.2007.03.001.

[12] J.M.D.L. Patacas, N. Dawood, M. Kassem, Evaluation of ifc and cobie as data sources for asset register creation and service life planning, in: 14th International Conference on Construction Applications of Virtual Reality, 2014, ISBN 978-0-9927161-1-0.

[13] C. Fu, G. Aouad, A. Lee, A. Mashall-Ponting, S. Wu, Ifc model viewer to support nd model application, Autom. Constr. 15 (2) (2006) 178–185, https://doi.org/10.1016/j.autcon.2005.04.002.

[14] buildingSMART, What Is ifc? accessed: 26/10/2020. URL in: https://technical.buildingsmart.org/standards/ifc/, 2018.

[15] Y. Adachi, Overview of ifc model server framework, in: European Conference on Product and Process Modelling (ECPPM), 2002, 367–372ISBN: 905809507X.

[16] T. Froese, F. Grobler, J. Ritzenthaler, K. Yu, B. Akinci, R. Akbas, B. Koo, A. Barron, J.C. Kunz, Industry foundation classes for project management-a trial implementation, ITcon 4 (Nov) (1999) 17–36.

[17] J.K.W. Wong, J. Ge, S.X. He, Digitisation in facilities management: a literature review and future research directions, Autom. Constr. 92 (2018) 312–326, https://doi.org/10.1016/j.autcon.2018.04.006.

[18] J. Heaton, A.K. Parlikad, J. Schooling, Design and development of bim models to support operations and maintenance, Comput. Ind. 111 (2019) 172–186, https://doi.org/10.1016/j.compind.2019.08.001.

[19] R. Vieira, P. Carreira, P. Domingues, A.A. Costa, Supporting building automation systems in bim/ifc: reviewing the existing information gap, Eng. Constr. Archit. Manag. (2020), https://doi.org/10.1108/ECAM-07-2018-0294.

[20] W. Kastner, G. Neugschwandtner, S. Soucek, H.M. Newman, Communication systems for building automation and control, Proc. IEEE 93 (6) (2005) 1178–1203, https://doi.org/10.1109/JPROC.2005.849726.

[21] B. Dave, A. Buda, A. Nurminen, K. Främling, A framework for integrating bim and iot through open standards, Autom. Constr. 95 (2018) 35–45, https://doi.org/10.1016/j.autcon.2018.07.022.

[22] M. Jia, A. Komeily, Y. Wang, R.S. Srinivasan, Adopting internet of things for the development of smart buildings: a review of enabling technologies and applications, Autom. Constr. 101 (2019) 111–126, j.autcon.2019.01.023.

[23] B.W. Boehm, Software engineering economics, IEEE Trans. Softw. Eng. 1 (1984) 4–21.

[24] J.C. Westland, The cost of errors in software development: evidence from industry, J. Syst. Softw. 62 (1) (2002) 1–9.

[25] O. Pastor, J.C. Molina, Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling, Springer Science & Business Media, 2007.

[26] J. Won, G. Lee, How to tell if a bim project is successful: A goal-driven approach, Autom. Constr. 69 (2016) 34–43.

[27] N. Ham, S. Moon, J.-H. Kim, J.-J. Kim, Economic analysis of design errors in bim-based high-rise construction projects: case study of haeundae l project, J. Constr. Eng. Manag. 144 (6) (2018) 05018006.

[28] S. Sun, L. Lannom, B. Boesch, Handle System Overview, Tech. rep., Internet Engineering Task Force, rFC 3650, Accessed: 26/10/2020. 2003, https://doi.org/

10.17487/RFC3650.
[29] L. Sabol, BIM Technology for FM, Vol. 1 of BIM for Facility Managers, John Wiley & Sons, New Jersey, 2013, https://doi.org/10.1002/9781119572633.ch2, iSBN: 978-1-118-38281-3.
[30] S. Tang, D.R. Shelden, C.M. Eastman, P. Pishdad-Bozorgi, X. Gao, A review of building information modeling (bim) and the internet of things (iot) devices integration: present status and future trends, Autom. Constr. 101 (2019) 127–139, https://doi.org/10.1016/j.autcon.2019.01.020.
[31] J. Park, J. Chen, Y.K. Cho, Self-corrective knowledge-based hybrid tracking system using bim and multimodal sensors, Adv. Eng. Inform. 32 (2017) 126–138, https://doi.org/10.1016/j.aei.2017.02.001.
[32] A. Kiani, A. Salman, Z. Riaz, Real-time environmental monitoring, visualization, and notification system for construction h&s management, J. Inform. Technol. Construct. 19 (2014) 72–91.
[33] S. Peng, G. Su, J. Chen, P. Du, Design of an iot-bim-gis based risk management system for hospital basic operation, in: IEEE Symposium on Service-Oriented System Engineering (SOSE), IEEE, 2017, pp. 69–74, https://doi.org/10.1109/SOSE.2017.22.
[34] J. Lee, Y. Jeong, Y.-S. Oh, J.-C. Lee, N. Ahn, J. Lee, S.-H. Yoon, An integrated approach to intelligent urban facilities management for real-time emergency response, Autom. Constr. 30 (2013) 256–264, https://doi.org/10.1016/j.autcon.2012.11.008.
[35] M. Marzouk, A. Abdelaty, Monitoring thermal comfort in subways using building information modeling, Energy Build. 84 (2014) 252–257, https://doi.org/10.1016/j.enbuild.2014.08.006.
[36] X. Yuan, C.J. Anumba, M.K. Parfitt, Cyber-physical systems for temporary structure monitoring, Autom. Constr. 66 (2016) 1–14, https://doi.org/10.1007/978-3-030-41560-0_7.
[37] R.Y. Zhong, Y. Peng, F. Xue, J. Fang, W. Zou, H. Luo, S.T. Ng, W. Lu, G.Q. Shen, G.Q. Huang, Prefabricated construction enabled by the internet-of-things, Autom. Constr. 76 (2017) 59–70, j.autcon.2017.01.006.
[38] W. Mazairac, J. Beetz, Bimql–an open query language for building information models, Adv. Eng. Inform. 27 (4) (2013) 444–456, https://doi.org/10.1016/j.aei.2013.06.001.
[39] M. Dibley, H. Li, Y. Rezgui, J. Miles, An ontology framework for intelligent sensor-based building monitoring, Autom. Constr. 28 (2012) 1–14, https://doi.org/10.1016/j.autcon.2012.05.018.
[40] S. Hu, E. Corry, E. Curry, W.J. Turner, J. O'Donnell, Building performance optimisation: a hybrid architecture for the integration of contextual information and time-series data, Autom. Constr. 70 (2016) 51–61, https://doi.org/10.1016/j.autcon.2016.05.018.
[41] S. Tang, D.R. Shelden, C.M. Eastman, P. Pishdad-Bozorgi, X. Gao, Bim assisted building automation system information exchange using bacnet and ifc, Autom. Constr. 110 (2020) 103049, https://doi.org/10.1016/j.autcon.2019.103049.
[42] N. Kahani, M. Bagherzadeh, J.R. Cordy, J. Dingel, D. Varró, Survey and classification of model transformation tools, Softw. Syst. Model. 18 (4) (2019) 2361–2397, https://doi.org/10.1007/s10270-018-0665-6.
[43] M.J. Pratt, Introduction to iso 10303—the step standard for product data exchange, J. Comput. Inf. Sci. Eng. 1 (1) (2001) 102–103, https://doi.org/10.1115/1.1354995.
[44] A. Magruk, The most important aspects of uncertainty in the internet of things field–context of smart buildings, Proc. Eng. 122 (2015) 220–227, https://doi.org/10.1016/j.proeng.2015.10.028.
[45] E. Oriwoh, P. Sant, G. Epiphaniou, Guidelines for internet of things deployment approaches–the thing commandments, Proc. Comput. Sci. 21 (2013) 122–131, https://doi.org/10.1016/j.procs.2013.09.018.
[46] K. Mekki, E. Bajic, F. Chaxel, F. Meyer, A comparative study of lpwan technologies for large-scale iot deployment, ICT Express 5 (1) (2019) 1–7, https://doi.org/10.1016/j.icte.2017.12.005.
[47] Z.-K. Zhang, M.C.Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, S. Shieh, Iot security: ongoing challenges and research opportunities, in: 7th IEEE International Conference on Service-Oriented Computing and Applications, IEEE, 2014, pp. 230–234, https://doi.org/10.13140/RG.2.2.27063.47520.
[48] P. Mohagheghi, W. Gilani, A. Stefanescu, M.A. Fernandez, An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases, Empir. Softw. Eng. 18 (1) (2013) 89–116, https://doi.org/10.1007/s10664-012-9196-x.
[49] N. Sornin, M. Luis, T. Eirich, T. Kramp, O. Hersent, Lorawan specification 1.1, accessed: 26/10/2020, URL in: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.
[50] M. Rizzi, P. Ferrari, A. Flammini, E. Sisinni, Evaluation of the iot lorawan solution for distributed measurement applications, IEEE Trans. Instrum. Meas. 66 (12) (2017) 3340–3349, https://doi.org/10.1109/TIM.2017.2746378.
[51] P. Spachos, I. Papapanagiotou, K.N. Plataniotis, Microlocation for smart buildings in the era of the internet of things: a survey of technologies, techniques, and approaches, IEEE Signal Process. Mag. 35 (5) (2018) 140–152, https://doi.org/10.1109/MSP.2018.2846804.
[52] X. Su, H. Zhang, J. Riekki, A. Keränen, J.K. Nurminen, L. Du, Connecting iot sensors to knowledge-based systems by transforming senml to rdf, Proc. Comput. Sci. 32 (2014) 215–222, https://doi.org/10.1016/j.procs.2014.05.417.
[53] A. Ruiz-Zafra, M. Noguera, K. Benghazi, S.F. Ochoa, A model-driven approach for wearable systems developments, Int. J. Distributed Sensor Networks 11 (10) (2015) 637130, https://doi.org/10.1155/2015/637130.
[54] R. Kahn, R. Wilensky, A framework for distributed digital object services, Int. J. Digit. Libr. 6 (2) (2006) 115–123, https://doi.org/10.1007/s00799-005-0128-x.
[55] P.T. Kirstein, A. Ruiz-Zafra, Use of templates and the handle for large-scale provision of security and iot in the built environment, 2018, https://doi.org/10.1049/cp.2018.0029.
[56] M. Pasetti, P. Ferrari, D.R.C. Silva, I. Silva, E. Sisinni, On the use of lorawan for the monitoring and control of distributed energy resources in a smart campus, Appl. Sci. 10 (1) (2020) 320, https://doi.org/10.3390/app10010320.
[57] A.M. Yousuf, E.M. Rochester, M. Ghaderi, A low-cost lorawan testbed for iot: Implementation and measurements, in: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), 2018, pp. 361–366, https://doi.org/10.1109/WF-IoT.2018.8355180.
[58] W. Li, G. Shen, J. Zhang, An indoor environmental monitoring system for large buildings based on lorawan, in: Proceedings of the Conference on Research in Adaptive and Convergent Systems, RACS '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 34–38, https://doi.org/10.1145/3338840.3355667.
[59] A. Banks, R. Gupta, Mqtt version 3.1. 1, accessed: 26/10/2020. URL http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf, 2014.
[60] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, 2nd edition, Addison-Wesley Longman Publishing Co., Inc., USA, 2003, https://doi.org/10.5555/773239.
[61] M. Leo, F. Battisti, M. Carli, A. Neri, A federated architecture approach for internet of things security, in: 2014 Euro Med Telco Conference (EMTC), 2014, pp. 1–5, https://doi.org/10.1109/EMTC.2014.6996632.
[62] S. Auer, H. Herre, A versioning and evolution framework for rdf knowledge bases, in: International Andrei Ershov Memorial Conference on Perspectives of System Informatics, Springer, 2006, pp. 55–69, https://doi.org/10.1007/978-3-540-70881-0_8.
[63] M. Weise, T. Liebich, J. Wix, Integrating Use Case Definitions for ifc Developments, eWork and eBusiness in Architecture and Construction, Taylor & Francis Group, London, 2009, pp. 637–645, https://doi.org/10.1201/9780203883327.ch71.
[64] M. Zhiliang, W. Zhenhua, S. Wu, L. Zhe, Application and extension of the ifc standard in construction cost estimating for tendering in china, Autom. Constr. 20 (2) (2011) 196–204, j.autcon.2010.09.017.