230

The Use of Templates and The Handle System for Large-Scale Provision of Security and IoT in the Built Environment

[redacted]

Peter T. Kirstein, Angel Ruiz-Zafra

University College London, United Kingdom, {p.kirstein, a.ruiz-zafra}@ucl.ac.uk

Keywords: IoT, BIM, Security, Handle System, Digital Objects.

Abstract

Building Information Modeling (BIM) is the main standard for constructions in the Built Environment. However, the currently agreed BIM interchange standards (IFC, COBie) are not suited for much further extension, lacking features needed for the incorporation of IoT and security. In addition, these standards are used to describe buildings and other assets through files, where the different building assets are entities with a fixed data structure and no relations between them. That is, two assets represented in the file, even described with the same data structure, are treated as separated elements, so that changes in one element do not affect the data structure of the other one. This will cause functional problems in future Building Automation Systems, such as the lack of provision for handling all the doors of a specific type at the same time when adding specific or general security features. To address these issues (lack of IoT and security in BIM and the independent and fixed data structure used to represent built assets), we present in this paper a novel approach, which is part of our research work in the EBIS project (Extending BIM Level 2 to support IoT & Security). The approach is based in the use of digital objects (DOs), instead of files, to represent built environments with IoT and security features. Also, these DOs could be used to define customised hierarchical structures to represent flexible data structures and relate built assets between them, what we called *templates*. These DOs to represent assets are supported by the Handle System, a Secure Identity Data Management System (SIDMS). This approach will allow the incorporation of generic and specific security features to all assets of a specific class. It will allow also the incorporation of assets defined generically in other domains to become templates, after appropriate processing, in the BIM databases. In order to validate our proposal, a Proof of Concept (PoC) is conducted in this research. We conclude with some comments on future work in the advertising of building features in a catalogue system – working in collaboration with our approach.

1 Introduction

The construction of a building is a long process that involves activities such as design or planning, cost estimation, assets, etc. This process is evolving, and new solutions are emerging to replace classical approaches in the built environment, as for example BIM (Building Information Modeling): an approach to address the lifecycle of a building, supporting the different activities involved through the use of models [1-3].

The models used in BIM, called BIMs (Building Information Models), are digital descriptions of every aspect of the built assets. To encourage wide adoption of this technology, there has been a broad international standardisation effort in the BIM activity. Indeed, the UK government has mandated its use since 2016. The actual standardisation so far has only progressed to BIM Level-2. Here, the interested parties have been able to agree interchange standards at the level of files (IFC [4], Industry Foundation Classes) and spreadsheets (COBie [5], Construction-Operations Building information exchange) that are quite inadequate for current requirements. Their use is not suitable to address the deployment of novel smart built environments (e.g. smart home, smart cities, smart buildings), because three main drawbacks: lack of IoT features, limited security capability and fixed asset data structures.

The Internet of Things (IoT) is a promising topic nowadays applied in several areas [6-9], including the built environments [10]. IoT has several benefits [11], but also several challenges, with ~~being the~~ security one of the major ones [12].

230

Current BIM models (IFC, COBie) and current BIM version (BIM Level-2) do not support IoT and security features, that is, the standards specifications are not ready to address the design of a smart built environments with IoT and security from early stages of the building design. This used to be added at a later stage through a Building Automation System (BAS) [13].

In addition, the different built assets described in the models (e.g. door, wall, light, chair, table) are described with a fixed data structure, that is, all the assets of a same category (e.g. door) are represented in the same manner, with the same data structure and regardless of the scenario or requirements.

In a novel smart built environment this is an issue, because in a real IoT scenario there are assets that belongs to the same category but with different features or objectives. For instance, in a smart building, there are different types of doors, classic doors, doors with RFID-based badge readers, smart door controlled by the building manager, etc., where each one works in a different way, is composed by different elements and, therefore, is described with a different structure.

In this paper we present an integrated solution to support IoT and security features, and the possibility to use dynamic asset data structures in built environments. These solutions are part of the EBIS (Extending BIM Level-2 to support IoT & Security) project, where we propose a framework to address the deployment of IoT scenarios in the built environments from scratch, providing the procedures to be followed as well as the software and techniques required.

To address the problem of lack of IoT and security, the proposal presented in this research focuses on the idea to give a hierarchic identity to each physical component. That is, create a digital representation (digital object -DO) to each physical asset, to hold the original attribute data in a secured database, where new IoT and security attributes can be added to the digital representation. The DOs will be supported by The Handle system, a Secure Identity Data Management System (SIDMS).

About the fixed asset data structure described in the building models, we introduce the concept of *template*, a customized hierarchical structure to support any physical asset structure or data description through digital objects supported by the Handle system and inheriting their security aspects.

In order to validate our proposal in this work, a Proof of Concept (PoC) in a built environment (smart building) is already presented in the paper.

This paper is organized as follows. Section 2 introduces the Handle system, describing the architecture and showing their main features and benefits. Section 3 shows how the Handle system can be used to support IoT and security features. Section 4 introduces the template concept, describing the model as well as the real implementation supported by the Handle system. Section 5 shows the PoC and, finally, Section 6 summarizes with the conclusions and the future work.

2 The Handle System

The Handle system is a comprehensive system for assigning, managing, and resolving persistent identifiers for digital objects on the Internet [14,15]. The features mentioned here are those included in the current implementation, version 8.1.

The Handle system is not just a database to store information and represent digital objects accessed via an identifier, also it includes an open set of protocols, an identifier space and an implementation of the protocols. These protocols enable a distributed computer system with two different levels: Global Handle Registry (GHR), managed by their developers (CNRI) on behalf of the international DONA foundation [16], and Local Handle Services (LHS), managed by individual administrators, corporations, universities, etc.

In Handle, a digital object identifier is composed of two different parts: a prefix and a custom (and unique) name called suffix, separated by the character "/", that is, prefix/suffix. For instance, handle "1234/handle1" is defined under a LHS with the prefix 1234 and has the unique name "handle1", as suffix.

230

Each digital object is composed of a set of equal data structures known as indices, where each index is identified by an integer number. Usually, each one is used for a specific purpose, according to the type.

The Handle architecture provides several features, such as scalability, extensibility, replicated storage (will run across as many computers as are required to provide the desired functionality) and performance (providing caching and hashing), among others [15]. However, one of the most useful features provided by Handle is the security.

The Handle system provides two different mechanisms to ensure the security of the different digital objects: Authentication and Authorisation. The authentication is the process to be identified as a digital object, while the authorisation is the possibility to be able to create, read, update or delete (CRUD) a specific digital object.

The authorisation is automatically managed by the Handle system, so if a user (identified as a specific handle) is not authorised to perform a specific operation, because is not authorised, The Handle system returns a non-authorised message. The authentication to be identified as a handle (using secret key or public/private key methods), is done through the protocols implemented by Handle: Basic Authorisation and Challenge-Response method [17]. Both protocols can be used directly through the Handle Software (in Java) or using the REST API provided by The Handle system, as the documentation shows [15].

These features provide several benefits and allow many possibilities in the goals presented in this paper: provides IoT and security features in built environments, using The Handle system as support for the different IoT scenarios (IoT applications, end-users, IoT devices, sensitive information, etc).

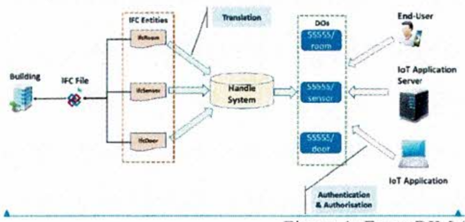
3 IoT and security through digital objects

Built environments deployed using BIM technologies have a lack of IoT and security. BIM technologies as IFC or COBie only support classical assets of a building such as doors, walls or lights, but are not able to represent IoT and security aspects, as for example, an IoT device which required a specific security token to be accessed or set who is authorised to access to a specific space (e.g. room).

To address this issue, the use of digital objects (DOs) is a promising approach, where the different built assets related to IoT are represented as digital objects. The Handle system provides the technological support to reach this approach, adding, among others, security constrains to the DOs.

Using the Handle System it is possible to represent built assets, store sensitive information and, through the security mechanisms (Authentication and Authorisation), ensure that only authenticated and authorised users (or digital objects) can access to the DO. In this way, it is possible, for instance, to define who is authorised to get access to a room or get the temperature value of a sensor hosted in an IoT device.

Figure 1 shows how with BIM technologies (IFC) we are able to represent a built environment using the Handle system, where each digital object could be accessed by users, IoT applications or application servers, through the security mechanisms provided by Handle.



[redacted]

Figure 1: From BIM (IFC) to the Handle System.

230

However, the use of the Handle system to represent IoT & security from BIM technologies has, at least, two main challenges:

1. Translation process from BIM (IFC) to Digital Objects (the Handle system)
2. Representation of IoT devices & security not specified in IFC

IFC files are composed by a set of entities (one per line) to define the different assets of a built environment, their relationships and features. Not all the entities of the IFC file are required in an IoT scenario, only the sensitive ones, that is, only those which should be considered part of an IoT scenario (e.g. sensor, smart door, an entire floor where are deployed IoT devices, restricted area accessed by a RFID-based door, etc.).

The direct translation process from sensitive IFC entities is carry out by a user using the tools provided by the Handle system, however, this is a slow process, because the user should create one by one the different DOs (reading the IFC file). In addition, the credentials, security tokens, identifier, etc. should be defined manually, which can cause data error or even security gaps.

In order to improve this task, and as a part of our research and the EBIS project, we propose an automatic tool called *IFC2Handle*: a cross platform application (Windows, Linux, MacOS) which uses the REST API provided by the Handle system to transform an IFC file into digital objects supported by Handle.

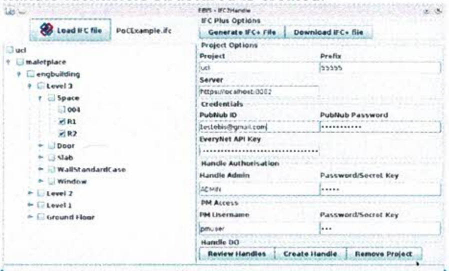
Through this application, an IFC file is used as input data. The application parses the IFC file and generates a checkbox tree structure, representing the structure of the building organized by levels, spaces, rooms, etc. The user selects (checking the checkboxes) the different parts of the building that he/she wants to be a sensitive part. That is, which parts of the building should be translated into digital objects (e.g. a specific door in a specific floor, an entire floor, etc).

With the selected parts of the building, and an additional information required to create the Handle structure (project name, prefix, Project Manager username and password, administrator credentials to be authorised, etc), the application generates automatically the Handle structure, creating all the handles (DOs) as well as configuring the authentication and authorisation of each handle.

IFC2Handle constructs each handle identifier automatically, according to the building levels (hierarchical) using a URL structure. For instance, if in the building model (IFC file) there is a sensor called *sensor1*, hosted in the *room1* of the *level3* (third floor), all under the project name *ucl*, and using the prefix *55555*, the handle identifier will be 55555/ucl/level3/room1/sensor1. It is this routine that must be replaced if the original BIM information is provided in a form different from IFC.

Figure 2 (a) shows the main screen of the IFC2Handle application and Figure 2 (b) shows an example about how the handle identifiers structured is created.

[redacted]



PoCExample.ifc

ucl
maletplace
engbuilding
Level 3
Space
004
☑ R1
☑ R2
Door
Slab
WallStandardCase
Window
Level 2
Level 1
Ground Floor

IFC Plus Options

Project Options
Project: ucl
Prefix: 55555
Server: https://localhost:8002

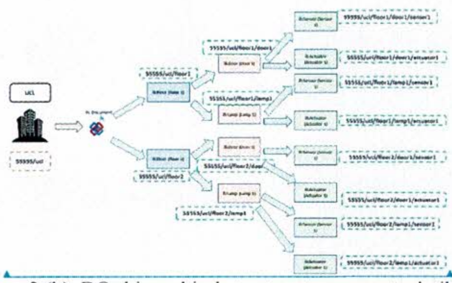
Credentials
PubNub ID: testebis@gmail.com
PubNub Password: ••••••••••••
EveryNet API Key: ••••••••••••••••••••••••••••••••

Handle Authorisation
Handle Admin: ADMIN
Password/Secret Key: •••••

PM Access
PM Username: pmuser
Password/Secret Key: •••

Handle DO

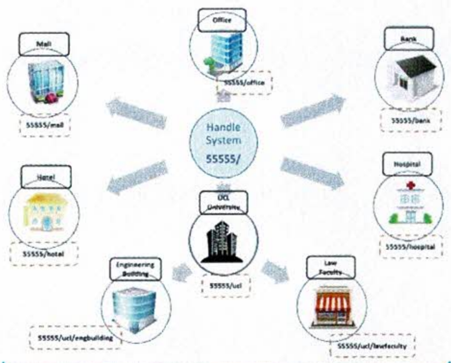
Figure 2 (a): IFC2Handle main screen



[redacted]

Figure 2 (b): DOs hierarchical structure to represent a building

The possibility to define a specific Handle structure under a prefix and a project name allows the possibility to support different IoT scenarios (e.g. different buildings) under the same Handle prefix (Handle installation) but with different credentials and users depending of the project (Figure 3).



[redacted]

Figure 3: Handle Structure to support different IoT scenarios

This way to support IoT scenarios for built-environments, through a hierarchical digital object structure with security constraints has several benefits, such as support of large-scale built environments, well-identified IoT elements, different types of IoT elements according to the type of the DO, etc.

However, this Handle structure created through the IFC2Handle application is just a first version of the DO structure to represent the IoT scenario, because new DOs should be added to support new entities required, such as end-users, additional security tokens, additional building assets, etc. as well as elements which are not supported by IFC.

To address this issue, a new application has been developed as a part of our work and the EBIS project. *IoTW* platform is a web platform to manage IoT scenarios represented by DOs supported by the Handle system.

Through IoTW platform, the user(s) are able to create new DOs to represent new IoT elements (IoT devices, sensors, actuators, security tokens, data fields), define the credentials (who is authorised to access to a DO), define new users (also represented as DOs), etc.
Figure 4 shows a screen of IoTW platform, which represent the completion process of an empty room. New IoT elements, not supported by IFC (e.g. end-node LoRa device, sensor with security token, etc), are added through this tool, creating new DOs.

230

[redacted]

Figure 4: Creation of new DOs to represent built assets

Each new DO created could be accessed through the REST API provided by The Handle system, according to the authentication and authorisation methods described in the official documentation [15]. In this way, different IoT applications of the IoT scenario or built environment are able to access to these DOs, in order to obtain relevant information or the security tokens to access to the real IoT devices.

4 Templates to support flexible data structures

4.1 Introduction

In a built environment there are many assets, such as doors, walls, lights, chairs, tables, etc. However, not all the assets of the same type are equals. For instance, in a built environment, there are classic doors, RFID-based doors, doors to enter to a public space, doors to enter to a private area, exit doors, etc.

In IFC, each entity to represent a specific asset has a static data structure to describe it, according to the IFC specification (id, owner, history, description, etc). Although could be possible add additional attributes to an IFC entity, two IFC entities of the same type with the same additional attributes are considered non-related entities, and, when they are translated to DOs, there are two different DOs without any relations, so changes in the data structure in one of them, do not affect the structure of the other one.

Furthermore, the absence of relation between the entities avoid the definition of a hierarchical structure with super types and subtypes, in order to be able to manage groups of entities at the same time, or apply changes in all the assets of the same type.

As a part of our work and the EBIS project, we have defined an approach based on a hierarchical structure of digital objects to represent dynamic data structures called *templates*. These digital objects are supported by the Handle system, inheriting their security features

4.2 Template description

In the EBIS project, a *template* is a custom and flexible data structure that includes a set of attributes, where each attribute is composed by the pair key-value (Hash table), where the key is the identifier (or name) of the attribute and the value is the data type. Each identifier of the data structure is unique, as well as the template name, to be uniquely referenced.

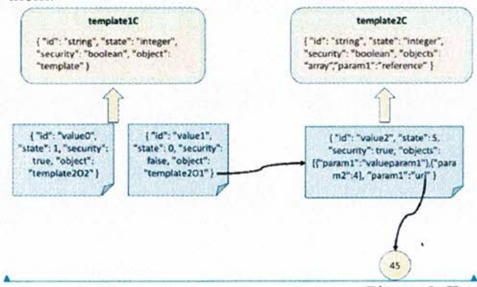
The different data types supported in our template approach are the primitive data types (string-char, integer, float, boolean), a recursive array of attributes (using the tag *array*), a reference to a different template (using the tag *template*) and a reference where the value can be stored (using the tag *reference*).

The template specification is structured using JSON notation and following the JSON specification [18], where types are restricted to the following tags: *string, integer, float, boolean, array, reference and template*.

230

This way to understand the templates allows the possibility to define any item using attributes, using primitive datatypes or the custom defined (*array, template, reference*). If the object contains ~~is composed by~~ an attribute composed by several objects, the array tag can be used. If the future value of an attribute is stored using a different template, the template tag can be used and, finally, if the value is stored in a different place (e.g. server, file, database), an URL can be setted using the reference tag.

Understanding a template as a data structure without any values, from a single template can exist several instances with different values. Figure 5 shows an example, where two different templates have instances, related between them.

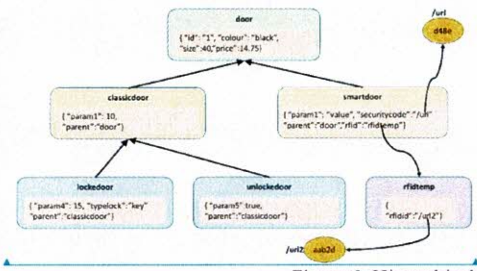

template1C
{ "id": "string", "state": "integer", "security": "boolean", "object": "template" }
template2C
{ "id": "string", "state": "integer", "security": "boolean", "objects": "array","param1":"reference" }
{ "id": "value0", "state": 1, "security": true, "object": "template2O2" }
{ "id": "value1", "state": 0, "security": false, "object": "template2O1" }
{ "id": "value2", "state": 5, "security": true, "objects": [{"param1":"valueparam1"},{"param2":4], "param1":"url" }
45

[redacted]

[redacted]

Figure 5: Templates instances

Finally, following this proposal, where a template can reference a different template, it is possible to construct a hierarchical organization, where a set of templates (sub-templates) can be under a template (super-template). In this way, a set of templates can be identified by the same type, but with differences according to the templates values or data structure. Figure 6 shows an example of a hierarchical organization/structure using the concept of "door" and templates instances (*classicdoor, smartdoor, lockeddoor and unlockeddoor*).


door
{ "id": "1", "colour": "black", "size":40,"price":14.75}
/url
d48e
classicdoor
{ "param1": 10, "parent":"door"}
smartdoor
{ "param1": "value", "securitycode":"/url" "parent":"door","rfid":"rfidtemp"}
lockedoor
{ "param4": 15, "typelock":"key" "parent":"classicdoor"}
unlockedoor
{ "param5" true, "parent":"classicdoor"}
rfidtemp
{ "rfidid":"/url2"}
/url2
aab2d

[redacted]

[redacted]

Figure 6: Hierarchical structure using templates

Template model has been designed to be malleable, so, new tags or changes in the data structure (e.g. add metadata to each template) could be added to support new requirements and cover useful functionalities.

4.3 Templates as digital objects supported by the Handle System

The current Handle system official release version supports templates, but with a different meaning. In that~~e~~ Handle System Release, a template is a handle (Digital object) created to solved full handles in a simple request, according to a specific pattern [15]. For instance, get URL values stored in different indices in the same handle in just one request, instead perform one request per each URL value (index).

In our work, as a part of the EBIS project, we have redefined a new template concept, according to the description of the previous section. However, the conceptual template description presented in the previous section is a generic

230

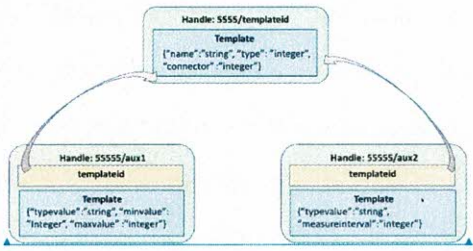
approach, and therefore, should be implemented to have a real (and ready to use) solution which can be applied to any domain, not just in the EBIS related area (BIM). To achieve a real implementation, the generic approach explained in Section 4.2 has been implemented using the Handle system.

In this implementation, each template is represented by a handle (Digital Object), identified by an identifier (prefix/suffix), where the data structure (JSON) is stored properly in the DO and with their own ACL (Access Control List). The use of ACLs can guarantee, for instance, that only authorised users are able to use or modify (read and write) the template. In some occasions, it could be useful to describe and publish templates only accessed by some types of users. For instance, all the templates related to a "sensor", could be accessed by the maintenance engineer of the building.

The representation of a template as a digital object (handle) ensures that two templates cannot have the same identifier, and a template cannot have two attributes with the same identifier. This guarantees that each template is unique, although can be used several times (sub-templates), but ensuring that if the template changes, the sub-templates should be modified according to the new changes.

Following with the description of Section 4.1, the tag *template* is replaced here using a specific index inside the handle, to specify the parent of the current template, using the name *parent*. The tag *reference* is known in this implementation with name *handle*, and can contains a URL, a database or even a handle identifier where the value is stored.

The use of handle identifiers as reference for other templates allows, like in the generic approach, the description of a hierarchical structure, where several templates (handles) are under one (or several) template (s). Figure 7 shows an example.


Handle: 5555/templateid
Template
{"name":"string", "type": "integer", "connector" :"integer"}
Handle: 55555/aux1
templateid
Template
{"typevalue":"string","minvalue": "Integer","maxvalue" :"integer"}
Handle: 55555/aux2
templateid
Template
{"typevalue":"string", "measureinterval":"integer"}

[redacted]

Figure 7: Use of The Handle system to represent template hierarchical structure

The templates implemented using handle have instances, that is, the data structure filled with values. These instances are represented as handles as well (DOs), where each template, stored in a handle, has an indeterminate (and unlimited) numbers of instances (handles) to store the values.

Furthermore, if the original template changes (e.g. add a new attribute), the handles that use the template (instances) also change, adding the new attribute. Although this option is not naturally supported by The Handle system, new services have been added to the original Handle Server to support it.

5 Proof of Concept

5.1 Background: EBIS Framework

The outcome of the EBIS project is the EBIS framework, designed to address, from scratch, the deployment of secure IoT scenarios for smart buildings.

230

The aim of the EBIS framework is, using BIM level-2 model(s) (IFC), deploy an entire IoT scenario for a smart building in the most automatic possible way with security constrains, where different IoT applications, IoT devices and users can interact between them.

So far, this PoC is based on the initial BIM data being expressed as an IFC file. However, we do not consider this a good basis, but it was the only one standardised to date. In the later APBIM work we will use a different form of that data such as databases, mainly because we expect two differences as a result of using a different form of the BIM data. First, there may be a richer set of types of component. Secondly, the routines for transforming the BIM data into databases accessible via Handle need to be rewritten.

The EBIS framework is composed by four different elements:

- A three-phases *methodology* to address the deployment of a multi stakeholder IoT scenario based on BIM.
- *Support technology* used to support the methodology proposed and the different processes through the three phases (The Handle system, Section 2).
- *Design elements* to enable IoT scenarios from BIM files. These elements are IFC+ (an extension of IFC to support IoT and security features) and templates (Section 4).
- *Support software* to improve and automate as much as it is possible all the processes related to the deployment of the system (IFC2Handle and IoTW, Section 3).

Through the methodology, the different users involved conduct the procedure to deploy the IoT scenario (using the support technology, design elements and support software), carrying out a set of tasks or processes, such as create the different DOs to represent the building, fill the values of the data structures and implements IoT applications, deploy sensor networks, etc.

The process starts with the creation of the first DO structure using IFC2Handle application (Section 3), and continues with the use of IoTW, where the different users complete the IoT scenario filling the different building assets (Section 3).

At this point, besides the representation of the different building assets as DOs through the IoTW platform, the users define the different templates. That is, define general templates to assign them to specific assets, according to the type, in order to categorize a set of assets of the same type and with the same data structure.

Because the description of the full PoC is out of the scope of this paper, in this section only will be addressed the part of the PoC related to the creation of the DOs structure (Section 3) and the templates (Section 4).

5.2 DO structure and templates in the EBIS PoC

The PoC defined to validate the EBIS framework is described as follows: There is a smart area inside of a building with two different rooms (R1 and R2) where the access is through R1. Each room have a temperature sensor, air conditioner machine as well as some lights, and the access is through a RFID badge reader hosted in the R1's door. In this scenario, we must define the different IoT elements, who is authorised to use them, who defines these authorisations, etc.

While the scale of this PoC is very small, it does exercise all the elements of the framework. The terms validate, and Proof of Concept are far-fetched, but at least this exercise both shows how the framework is applied and has shown up no insuperable bars to scaling up to more realistic scenarios.

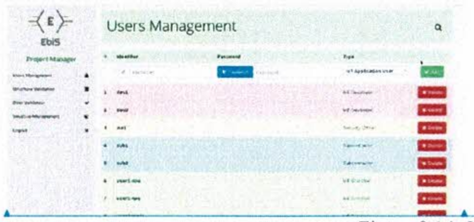
In this section, we describe only the first steps or processes of this PoC, where the creation of the DO structure and templates are involved.

First of all, the EBIS framework starts with the first phase of the methodology: Construction and Deployment. The architect, using an open-source or private vendor CAD software, design the building and generate an IFC file to represent the building or built environment.

230

With the IFC file, and using the IFC2Handle software already described in section 3 (Fig. 2 - a), the administrator of the system generates the first instance of the DO structure, following a URL path structure as Fig.2 (b) shows. In this case, and according to the PoC description, the administrator only selects the room number one and two (R1, R2) of the third floor, defined as *Space*, as sensitive parts of the building, and therefore, digital objects.

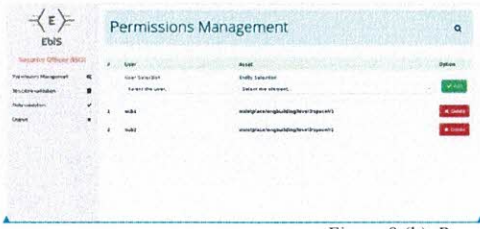
When the first instance of the DO structure is created (R1 and R2 defined as DO), the Project Manager (PM), defined through the IFC2Handle, use the IoTW platform (Figure 4) to define the rest of users involved in the PoC: Information Security Officer (ISO), Sub1 and Sub2 (subcontractor as architects, experts) to design R1 and R2, different end-user of the IoT applications and IoT developers (Fig. 8 – a).



[redacted]

[redacted]

Figure 8 (a): User Management



[redacted]

[redacted]

Figure 8 (b): Permissions management

The ISO can now login into the IoTW platform, and gives permissions to sub1 to manage R1 and sub2 to manage R2 (Fig. 8 – b).

At this point, the different subcontractors (sub1 and sub2) are able to:

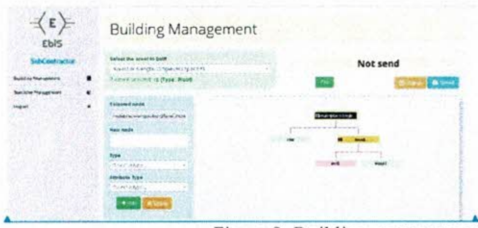
1. Complete the different empty spaces (R1 and R2) with different elements to represent the IoT scenario. That is, create new digital objects as digital representations of the physical assets in the room one and two.
2. Define the data structure of each element (DO) through the use of templates.

The creation of new DOs as well as the definition of templates is done through the IoTW platform, which provides a friendly interface using a three-based chart. Starting from the root, the subcontractor can add as many nodes (Children) as he/she needs, with several levels or deep (no limits) and where each node is defined by a name and a type.

In this PoC, the R1 contains a RFID badge reader and an End-Node LoRa device to open/close the door and manage the air conditioning machine. The user sub1 defines an end-node LoRa device, with the identifier *lora1* and a RFID badge reader, as a sensor, with the identifier *rfid*. In addition, the *lora1* element has several children, to represent the temperature sensor (with the identifier *temp1*) and an actuator used to calibrate several parameters (*act1*).

230

User sub1 can modify as many times as he/she needs the design of the room one, just clicking the "Save" button, being able to complete a difficult design during an undetermined period of time. Once the design is completed, sub1 click on the "Send" button. Figure 9 shows the view of the design of the R1 by the user sub1.



[redacted]

Figure 9: Building management by the user sub1 (editing Room 1)

When sub1 sends the design, this is now able to be evaluated by the PM (Project Manager) and is no longer available to be modified by the user sub1, showing the state "Not Send" to "Under Review". Sub2 do the same process to complete the R2.

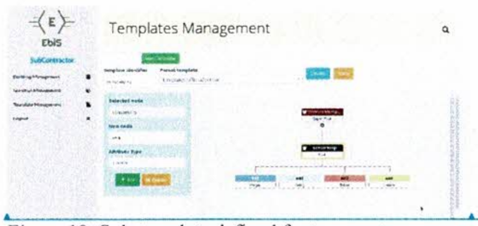
The PM access to their profile in the IoTW platform and click in the "Structure Validation" option. In this view, the PM selects, from a list, the different spaces to be validated.

In this case, there are two, related to the room one and room two. When the user clicks on each one of them, the PM can review the structure/design, through the same three-based chart used by the subcontractor. In the same view, the PM can "Accept" the proposal/design or "Reject" it. In case the PM accept the design, the platform makes a request to the Handle Server and create the different handles identifier according to the design. For instance, in this PoC, if it is accepted, the following handles are created (for room 1):

- 55555/ucl/maletplace/engbuilding/space/level3/r1/lora1
- 55555/ucl/maletplace/engbuilding/space/level3/r1/lora1/temp1
- 55555/ucl/maletplace/engbuilding/space/level3/r1/rfid

Once a space structure or design has been validated/accepted by the PM, the subcontractor, in their "Building Management" section, see their space structure as "Accepted". At this point, the subcontractor, is able to set the data structure (template) of each asset/part of the building.

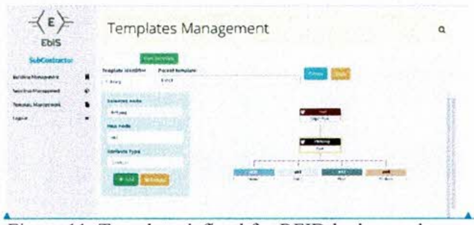
Although the data structure or template is setted in the "Building Management" section, the IoTW platform provides a dedicated section to manage the templates. In the "Templates Management", the subcontractors are able to define template structure from scratch (root node template) or from already defined templates (sub-templates). In this PoC, and for the user sub1, we define three new templates. Two of them are for the temperature sensor and the actuator, and are sub-templates of the template sensor. The other one is for the RFID badge reader, and there is a new template with the parent root, that is, is in the top of the hierarchical template structure. Figure 10 shows the definition of the templates related to the temperature sensor and Figure 11 shows the template to describe the RFID badge reader



[redacted]

Figure 10: Sub-template defined for temperature sensor

230



[redacted]

Figure 11: Template defined for RFID-badge reader

With the templates defined, the sub1 is able now to set this templates as data structures to define the different assets of the R1 (rfid, temp1 and act1).

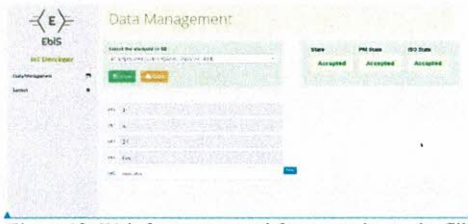
As in the previous step, the definition of a new building asset should be accepted by the PM (and sometimes by the ISO, if there are security attributes involved). The PM (and ISO), using their options in the IoTW platform, access to the ~~the~~ "Structure Validation" and check the different data structure (templates) of each building asset.

As a part of the second phase of the methodology (IoT applications applications), if the data structure is accepted, is now ready to be assigned to IoT developers, in order to fill each template or data structure with specific values (template instance).

The IoT developer (dev1), through the IoTW platform is able to define a template instance. The platform, automatically, generate a user- friendly -web form to represent the template and to be filled by the developer.

The developer can fill the form, save it and send it when is completed. The PM and ISO should accept the data values (sensitive data is encrypted and not shown). When the PM and ISO accept the different values, the different handles to represent the different attributes of the template, with their values, are created. These DOs, with the inherited security restrictions can be accessed through the REST API by the different IoT applications to support the required functionalities.

In the same way, if at any point, a new attribute is added in a super-template, the sub-templates inherit the attribute, and the user-friendly -forms change, showing a new attribute to be filled, creating a new DO to represent the new attribute with the new value (Fig. 12)



[redacted]

Figure 12: Web form generated from template to be filled by IoT developer

In this second phase, the different IoT applications for end-users are also implemented. When the second phase is completed, the third phase starts (Operations), where run the different application servers, IoT applications, exchange information between users and servers, users and IoT devices, get access from the sensors, etc

6 Conclusions and further work

BIM (Building Information Modeling) is the future in so far as it is the most information rich approach to planning and construction that we have nowadays. However, the Internet of Things (IoT), with its huge potential, is not supported in the BIM Level-2 Standard Specifications such as IFC and COBie or other parts of the standard. These aspects must be added at a later stage of the design or even construction.

To address this lack of IoT and security, in this paper we have proposed an approach based on the use of secured digital objects instead of files, to represent building assets. Each relevant building asset in a secure IoT scenario (e.g. private room, temperature sensor, RFID-badge reader, etc) is represented as a digital object (DO), supported by the Handle System. Each digital object has its own structure and security mechanisms (authentication, integrity and authorisation) and could be accessed by any type of device, anywhere and at any time from the REST API provided

In this way, it is possible to safeguard relevant information in the DOs and ensure that only authenticated and authorised users can get access. These DOs can be used to store security tokens, private and encrypted data, etc. At the same time, we must stress that this approach provides only a technology base; Part 5 of the BIM Level-2 Standard [19] stresses that the large-scale publishing of building properties is by itself potentially dangerous, particularly when combined with data published in other ways.

This technology does not address the resulting threats, though it may give tools for mitigating them. For example, in the PoC example above, there are two secure rooms. To publish the location of these rooms, even without revealing details on how they may be accessed, gives undesirable information to potential attackers. It may be much more desirable to put all sensitive information, including as much information as possible about how to find it, into areas that are not accessible at all to unauthorised entities.

Another aspect stressed in [19] is the need to run secure servers. This includes many aspects of their operating systems and operational environment. The security of the global decentralise~~d~~ servers ~~architecture~~ of the Handle System, as defined in Section 2, need not be of major concern. That allows access only to the local system, which can take its own precautions. However, vulnerabilities in the application-specific servers could endanger the security of the BIM data. Similarly use of external templates, if they run on vulnerable systems, could contain malware that must be carefully inspected prior to use. While this sort of consideration is vital for the real security of the BIM data, it is ignored in EBIS and other projects currently planned.

In the representation of a physical asset as a DO, there is a data structure store in the DO which is used to record the physical asset definition (e.g. an asset is described by a height, weight, colour, width, type, etc). This data structure is different depending of the type of asset, because two different assets are described in a different way. (e.g. a classic door and one equipped with an RFID badge reader).

Although it is possible to describe different data structures into DOs, there are not related at all, so changes in a data structure do not affect other DO with the same data structure. To provide facilities allowing a closer link between the type of component and its basic structure, we have proposed in this paper the concept of template: a customised hierarchical structure to support any physical asset structure or data description of digital objects supported by the Handle system and inheriting their security aspects.

Both the representation of assets as DOs and the use of template for related data structure are part of the EBIS framework. To validate this framework, a Proof of Concept (PoC) has been carried out. In this paper, we have present the part of the PoC related to the goals this work, showing how it is possible to represent building information as DOs and use template to describe data structure.

We stated that validating the whole technology is a goal of EBIS. Clearly the range and extent of the PoC deployment too small to give a real validation. Although EBIS project shows promising and interesting results, it is not yet complete and several areas need further study. We have not yet considered seriously how to integrate in current building management systems. Also, although the Handle System is a very useful specification and support tool, it is difficult, for instance, discover the different DOs or identifiers in a building information database. That is, only it is possible access to a DO if you know their identifier.

230

It is not obvious how to list all the different building assets represented as DOs of a building, to know about its resources or elements. To address this issue, as future work, we are researching about the potential use of Hypercat. Hypercat is a standard driving secure and interoperable Internet of Things (IoT) for Industry. Of course, it is not really the Hypercat Standard that matters, but the actual implementations. For instance, one part of the Hypercat Standard (Section 7 of the official Hypercat documentation) defines security facilities in Hypercat. However, it gives a wide choice of algorithms and mechanisms, with no comment on what should be used. In any real use of Hypercat, it is the properties of its implementation that must be studied if one wishes to consider what should be done in the Hypercat servers and what in the Handle ones.

By a combined use of Hypercat, as well as the Handle System, it should be possible to publish all the resources of a building, with Handle identifiers, in order to be used by third-party software or users. Although Hypercat will list the different resources, the security is still supported by the Handle System.

Finally, part of our future work under APBIM and PETRAS auspices will extend the validation to work through the whole technology in the context of the deployment of a smart, instrumented home with provision for IoT and security. This work will be supported by the use of the EBIS framework with the collaboration of the Building Research Establishment (BRE), who will provide the real BIM data and access to their Innovation House

Acknowledgements

We acknowledge the support of the EPSRC and UCL Enterprise for their support for the EBIS project and EPSRC and PETRAS for their support for the APBIM project in the conduct of this research. The general technology and the concentration on IFC was part of EBIS; the IFC activity will not be pursued under APBIM.

References

[1] Eastman, C. M., Eastman, C., Teicholz, P., & Sacks, R. (2011). BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors. John Wiley & Sons.
[2] Azhar, S. (2011). Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. Leadership and management in engineering, 11(3), 241-252.
[3] Standard, P. A. S. 1192-3: 2014. Specification for information management for the operational phase of assets using building information modelling.
[4] Liebich, T. (2013). IFC4—The new buildingSMART standard. In IC Meeting. Helsinki, Finland: bSI Publications. ISO 690.
[5] East, W. (2007). COBIE (Construction-Operations Building Information Exchange). US Army Engineer Research and Development Center, US Army Corps of Engineers, Washington, DC, USA.
[6] Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. CISCO white paper, 1(2011), 1-11.
[7] Gubbi, J., Buyya, R., Marusic, S., &; Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.
[8] Atzori, L., Iera, A., &; Morabito, G. (2010). The internet of things: A survey. Computer networks, 54(15), 2787-2805.
[9] Sanchez, L., Muñoz, L., Galache, J. A., Sotres, P., Santana, J. R., Gutierrez, V., ... & Pfisterer, D. (2014). SmartSantander: IoT experimentation over a smart city testbed. Computer Networks, 61, 217-238.
[10] Jie, Y., Pei, J. Y., Jun, L., Yun, G., & Wei, X. (2013, June). Smart home system based on iot technologies. In Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on (pp. 1789-1791). IEEE. ISO 690.
[11] Manyika, J., Chui, M., Bisson, P., Woetzel, J., Dobbs, R., Bughin, J., & Aharon, D. (2015). Unlocking the Potential of the Internet of Things. McKinsey Global Institute.
[12] Lee, I., & Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons, 58(4), 431-440. ISO 690.
[13] Jung, M., Reinisch, C., & Kastner, W. (2012, July). Integrating building automation systems and ipv6 in the internet of things. In Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on (pp. 683-688). IEEE.

[14] 14. Sun, S., Lannom, L., & Boesch, B. (2003). The Handle system overview (No. RFC 3650).
[15] http://hdl.handle.net/20.1000/105
[16] https://www.dona.net /
[17] SWIFT, Michael M.; SHAH, Bharat. Challenge-response authentication and key exchange for a connectionless security protocol. U.S. Patent No 6,377,691, 23 Abr. 2002.
[18] https://tools.ietf.org/html/rfc7159
[19] BSI, "PAS 1192-5:2015 Specification for security-minded building information modelling, digital built environments and smart asset management," 2015.

[redacted]