A Digital Twin Platform Generating Knowledge Graphs for Construction Projects

Kyriakos Katsigarakis[1], Georgios N. Lilis[1], Dimitrios Rovas[1], Salvador González-Gerpe[2], Socorro Bernardos[2], Andrea Cimmino[2], María Poveda-Villalón[2] and Raúl García-Castro[2]

[1]*University College London, London, UK*
[2]*Ontology Engineering Group, Universidad Politécnica de Madrid, Spain*

Abstract

Construction projects combine activities across diverse domains and involve various entities which exchange information in different formats. To connect such diverse entities supporting their data exchanges using semantic web technologies, a Digital Twin Platform (DTP) is introduced, as a part of a greater ICT framework called COGITO, which aims at optimizing and supervising real construction projects from the conceptual to their implementation stages. To perform these connections, DTP creates a digital twin model designed to be the main data pool of COGITO's application tools. DTP's digital twin model is populated based on a well-defined ontology combining different data sources such as OpenBIM, time schedule, and construction resource files into a single RDF file. The digital twin model generation and access are demonstrated successfully on simple 4D OpenBIM data.

Keywords

Digital Twin, Ontology, IFC, OpenBIM

1. Introduction

The concept of Digital Twins (DTs), has emerged recently [1] as a digital entity that interacts with the digital ICT domain (digital cloud of World Wide Web), emulating the behavior of a physical counterpart (product or process, or system). According to [1], a Digital Twin has three main sub-components: physical product, virtual product, and the linkage between physical and virtual products. Since the physical entity of a Digital Twin is general enough to include physical systems consisting of products and processes on these products [2], it can be applied across Architecture Engineering Construction Operation and Facility Management domains (AECO-FM industry) [3, 4] which involve these entities.

As far as the construction domain is concerned, construction projects are complex processes involving products, time, scheduled processes, and resources. This application of the Digital Twin concept in construction projects led to the introduction of the Digital Twin Construction concept [5]. A Digital Twin in the construction domain is a concept that should not be

Third International Workshop On Semantic Digital Twins (SeDiT 2022), co-located with the 19th European Semantic Web Conference (ESWC 2022), Hersonissos, Greece - 29 May 2022

0000-0002-2748-4506 (K. Katsigarakis); 0000-0002-0642-5291 (G. N. Lilis); 0000-0002-5639-6783 (D. Rovas); 0000-0003-1550-0430 (S. González-Gerpe); 0000-0003-1790-5941 (S. Bernardos); 0000-0002-1823-4484 (A. Cimmino); 0000-0003-3587-0367 (M. Poveda-Villalón); 0000-0002-0421-452X (R. García-Castro)

confused with the concept of a building information model (BIM) [6] since according to the Gemini Principles [2] published by the Centre of Digital Build Britain (CDBB): a Digital Twin is a realistic digital representation of assets, processes, or systems in the built or natural environment. Therefore, the Digital Twin of a construction process asset might include the BIMs as a sub-component of the building assets if any. Recently, significant efforts to extend the use of openBIM standards such as IFC [7] to non-building elements such as roads [8], rail structures [9] and bridges [10] took place, facilitating the adoption of openBIM to construction projects and respective assets of Construction Digital Twins.

Aligned to these digitization efforts in the construction domain via the use of openBIM standards, a Digital Twin Platform (DTP) is introduced in the present work, as a central component of an ICT framework called COGITO, of a European H2020 project. COGITO aims to support construction projects in their conceptual and implementation stages. DTP integrates input data originating from the construction project stakeholders and installed IoT devices on the construction site, to provide information to COGITO's Data Quality checking Health & Safety provision, and Project Supervision tools. This data integration occurs in DTP's Knowledge Graph generator component (KGG), which is going to be the main topic of the present work. Internally, KGG generates a common data model structure, which will act as a data provider for all involved COGITO tools using semantic web-linked data technologies to connect COGITO's diverse input data. KGG essentially produces the semantically linked data model of COGITO, according to an ontology appropriately defined to cover all of COGITO's data requirements.

Similar attempts to support construction works using digital twins ICT deployments have also recently been initiated: ASHVIN project [11] focusing on general construction projects, SPHERE project aims to develop a Digital Twin supporting BIM [12].

Given the previous introduction, the rest of the paper is organized in a bottom-up fashion, starting from the description of COGITO's data structures, domains, and ontology, passing to the introduction of the DTP's layered architecture, and ending at the population and accessing of COGITO's digital twin model. A simple 4D BIM example is presented demonstrating the correct COGITO digital twin model creation and access.

2. Data structures

2.1. Data domains

COGITO's input data were classified into three domains based on their characteristics: Construction, Resource, and Process. Each of the three domains contains data from different sources. More specifically, these input data domains are:

1. **Construction domain (CONS)** which contains data referring to the construction site elements. The data are structured in an OpenBIM format following the latest IFC4.3 specification [7], which extends the building-related data of the previous IFC schema releases, to non-building-related infrastructures such as roads, bridges, and rail.
2. **Resource domain (RESO)** which contains the necessary data structures which define the resources of the construction project: human (workers and roles) and non-human (tools, equipment such as IoT devices and their data).

Table 1
List of ontologies created an reused in the COGITO project

Prefix	Namespace	Origin
beo	https://pi.pauwel.be/voc/buildingelement#	reused
bot	https://w3id.org/bot#	reused
facility	https://cogito.iot.linkeddata.es/def/facility#	created
geo	http://www.w3.org/2003/01/geo/wgs84_pos#	reused
process	https://cogito.iot.linkeddata.es/def/process#	created
qual	https://cogito.iot.linkeddata.es/def/quality#	created
resource	https://cogito.iot.linkeddata.es/def/resource#	created

3. **Process domain (PROC)** which includes all the construction schedule data referring to construction tasks involving construction resources and construction site elements.

Essentially, the *Process* domain contains data (scheduled tasks), which refer to the other two domains (*Construction*, *Resource*). To link the construction-related data across these diverse domains, an ontological scheme is defined, which reuses some existing ontologies and introduces new ones, where data gaps and missing relationships are identified. This new scheme is presented next.

2.2. COGITO Ontology

An ontology network consisting of four modules has been developed for the COGITO project. Three modules correspond to the three domains described in the previous section, and a fourth one has been created to capture quality. The prefixes and corresponding ontologies, created or reused in the COGITO ontology are listed in Table 1.

The complete and up-to-date documentation of each ontology module is provided online (See https://cogito.iot.linkeddata.es/). In this section, only the main concepts and modeling decisions are detailed.

1. **Facility Module** (corresponding to the construction domain). The classes and properties used to describe the topological concepts of a building in this module are based on the Building Topology Ontology (BOT), while other construction products, such as railways, bridges, and roads are described by new classes and properties.

    - `facility:Site` is defined as a subclass of `bot:Site` and, as such, a part of the physical world or a virtual world that is inherently both located in this world and having a 3D spatial extent. It contains (`bot:containsZone`) one or more `facility:Facility`.
    - `facility:SpatialZone` is defined as a subclass of `bot:Zone` and, as such, a part of the physical or a virtual world that is inherently both located in this world and has a 3D spatial extent. This class is the root of a hierarchy that includes `facility:ConstructionZone` -used to represent zones used by the health and Safety module- and `facilityTrackedZone` -used to represent zones used by the IoT preprocessing module.

- facility:Space is defined as a subclass of bot:Space and, as such, a part of the physical world or a virtual world whose 3D spatial extent is bounded actually or theoretically, and provides for certain functions within the zone it is contained in.
- facility:Facility is defined as something designed and built to serve a specific function affording a convenience or service. This new class was introduced to align the ontology to the new IFC4x3 extension to non-building elements and includes facility:Railway, facility:Bridge, facility:Road and facility:Building.
- facility:FacilityPart is defined as something that is contained (facility:hasFacilityPart) in a facility:Facility. A facility:FacilityPart can contain sub-facility parts (facility:hasSubFacilityPart).
- facility:Storey is defined as a subclass of bot:Storey and, as such, is contained (bot:hasStorey) in one facility:Building, and is intended to contain (bot:hasSpace) one or more facility:Space that are horizontally connected.
- facility:Element is defined as a subclass or bot:Element and, as such, constituent of a construction entity with a characteristic technical function, form, or position. A facility:Element can contain sub-elements (bot:hasSubElement). A facility:SpatialZone can contain (bot:containsElement) some facility:Elements (and so do facility:FacilityPart, facility:Space and facility:Storey). A beo:BuildingElement is a sub-class of facility:Element. This class involves a process:Task and there is information about its visual quality (qual:Defect) and geometric quality (qual:GQInf).

These BOT sub-classes have been created because they have specific properties such as facility:hasName and props:hasCompresedGuid. They are also sub-classes of geo:SpatialThing in order to reuse its location properties. There is a class shared among the four modules: facility:Project, which is defined as a large or major undertaking, especially one involving considerable money, personnel, and equipment. A facility:Project is related to a facility:Site and one or more process:Process, and each qual:Image and qual:PointCloud.

2. **Process Module** (corresponding to the process domain). The main classes and properties in this module have been defined anew since we could not find an appropriate ontology representing its concepts:

   - process:Process is defined as a series of actions aimed at accomplishing some result (in this case, related to a facility:Project).
   - process:Task is defined as a piece of work, which is carried out in a process:Process (process:belongsTo); and might be related to a facility:Element. A process:Task can have information about its duration (process:hasBeginningDate and hasEndDate).We can include the process:status and the process:progress of a process:Task. A task can be assigned (resource:hasResourceType) several ResourceType.
   - process:Cost is defined as the price paid to acquire, produce, accomplish, or maintain anything (in this case, process:Process and process:Task); and this price is measured (process:measuredIn) in a currency (process:UnitOfCurrency).

- process:WorkOrder is defined as a command or instruction authorizing specific work, repairs, etc., to be done. It has several resource:Resource assigned (resource:hasAssginedResource), one of which is a main provider (resource:hasMainProvider) and belongs to (process:belongToProcess) a process:Process.

3. **Resource Module** (corresponding to the resource domain). The main classes and properties of this module are the following:
    - resource:Resource is defined as a source of supply, support, or aid, especially one that can be readily drawn upon when needed. resource:HumanWorker, resource:Equipment and resource:trackingTrack are subclasses of resource:Resource. It is also a subclass of geo:SpatialThing in order to reuse its location properties. A resource:Resource belongs to a resource:ResourceType.
    - resource:ResourceType is defined as the kind of resources assigned to a process:Task or involved in a process:Process, indicating their maximum quantity (resource: maxUnit) and cost (resource:costPerHour)
    - resource:HumanWorker is defined as a laborer or employee who plays a role (resource:HumanRole) for a process:WorkOrder.
4. **Quality Module**. The main classes and properties of this module are the following:
    - qual:Defect is defined as a shortcoming, fault, or imperfection regarding a particular facility:Element. This qual:Defect is reflected is an qual:Image, which can be processed and is taken and a time and location on a kind of material.
    - qual:GeometricQualityInformation is defined as data informing of the a particular problem regarding a particular qual:Rule on a facility:Element. This information is part of a qual:ListOfGeometricQualityInformation that comes from analysing a qual:PointCloud.
    - qual:SafetyInformation is defined as data to prevent injuries by taking into account the characteristics of a facilityConstructionZone.

3. Digital Twin Platform

To support the creation of a semantically linked data model in the COGITO framework based on the ontology described in the previous section, a Digital Twin Platform (DTP) is designed and implemented. DTP acts as a data integration middleware, responsible for supporting appropriate data flows among COGITO tools and external data sources. The architecture of DTP consists of components belonging to the following six layers (displayed in the left part of Figure 1):

1. **Authentication Layer:** The Authentication Layer is responsible for storing and managing user accounts and their roles, enabling the DTP to register and authenticate the users of the COGITO platform.
2. **Data ingestion Layer:** The Data Ingestion Layer of DTP includes software components responsible for project creation, BIM data consistency validation, and COGITO

data model creation and semantic linkage. The structure of this layer is presented in the block diagram in the right part of Figure 1.

a) **Input Data Management** Routes input data traffic to other internal Data ingestion layer components (if it is BIM-related to the BIM management component if it is not BIM-related to the Knowledge Graph Generation component).
b) **BIM Management** It handles COGITO's input openBIM models that conform to the Industry Foundation Classes (IFC) standard [7] and performs initial operations in their data structures (serialization/deserialization and geometric data extraction).
c) **Knowledge Graph Generation** This component is responsible for generating, validating, storing, and linking RDF graph data according to the defined COGITO ontology. It contains the following three modules:
   - *Thing Manager* is responsible for routing the input data files inside the system according to the type of information handled or the desired operation mode. In addition, it is also the responsible for creating the WoT Thing Descriptions[1] (central building block in the W3C Web of Things, considered as the entry point of a Thing) referring to the data that belongs to the input data files, which will contain the information of the endpoints of the files, of the access to the RDF generated by the knowledge graph generator and of the different subjects described in the RDF.
   - *Knowledge Graph Enrichment* receives files from the Thing Manager and transforms them to RDF data according to the COGITO ontologies.
   - *RDF Graph Linker* is responsible for creating the connections between new and existing RDF data to generate a unified knowledge graph.
   - *RDF Data Validator* performs validation checks to ensure the generated RDF data from both the knowledge graph generator and graph linker, complies with the COGITO ontology, with no missing values or incorrect data types.

3. **Data Persistence layer:** The Data Persistence Layer of DTP contains different types of data stores for storing structured data: (a) a File storage system for storing files generated by COGITO applications, (b) a Project database for storing project- and user-related data, (c) a key-value database for storing IFC objects, (d) a time-series database, for storing IoT sensor data, (e) a triplestore for storing COGITO's knowledge graph(s) in the form of RDF files and (f) a thing directory for SPARQL translation.
4. **Data management layer:** The Data Management Layer of DTP is responsible for checking and routing the data queries of COGITO tools, ensuring that data have been correctly retrieved from the Persistence Layer and efficiently delivered to their destinations. It includes appropriate API wrappers to interface the datastores of the Persistence Layer (data providers) with the various COGITO applications (data consumers). Essentially, the data management layer has the necessary components which ensure correct, fast, and efficient response to hybrid queries referring to RDF- and not RDF-related data.
5. **Messaging layer:** DTPs' Messaging Layer is responsible for transmitting messaging data asynchronously between the Data Management Layer and the various COGITO applications.

[1] https://www.w3.org/TR/wot-thing-description/

6. **Data Post-processing layer:** The Data Post-processing Layer of DTP handles time-consuming IFC-related processes such as IFC optimization, MVD model checking, and IFC boundary representation (B-rep) generation for triangulated OBJ file exportation.

Using the aforementioned DTP infrastructure, COGITO data model creation and semantic linkage can be realized, as described in the following sections.

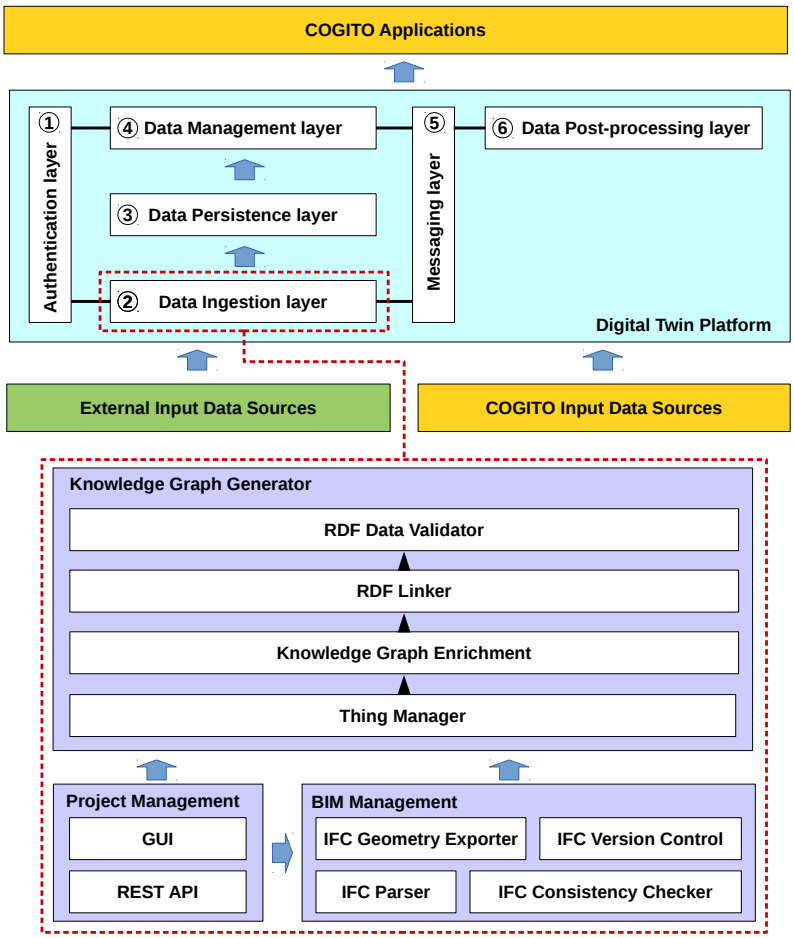

Figure 1: Architecture of DTP (top) and structure of DTP's data ingestion layer (bottom)

4. COGITO's digital twin model population

4.1. Operation sequence

Different DTP components are involved in the population of the COGITO ontology, each one having a different role. For this population a sequence of steps is followed including:

1. A COGITO user is being created via the platform GUI and its access is authenticated by the Authentication layer of DTP.
2. The user creates a project via the GUI and a specific ID is assigned to it by the project management component of DTP's Data Ingestion layer.
3. After the project is being created, the user uploads the data files via the platform's GUI related to that project. These files can be BIM data files (from the construction domain) and non-BIM data files (from the resource and process domains).
4. If the inserted files are BIM data files (openBIM data in the form of IFC files), they are directed to the BIM Management component of the data ingestion layer of DTP where a series of operations are performed, which include: version and consistency checks, deserialization, the load of the IFC objects on memory and extraction of geometric content.
5. The remaining input data structures of the inserted input data files, excluding analytic geometric descriptions and IoT data, are then forwarded to the Knowledge Graph generator, where they are converted into RDF data. For this conversion, numerous transformation engines are described as Thing Descriptions, which can be accessed dynamically depending on the type of file you want to transform to RDF. To know which translation engine to access, a preliminary check will be made to determine the type of file to be transformed. To perform this check, a query will be made to the Thing Directory (where the Thing Descriptions are stored), to access the Thing Description of the translation engine that performs the conversion to RDF of the specified file format, and therefore obtain the endpoint where the file will be sent for translation. In the case of transforming a BIM file, the specific ETL tool for these files will be used, but in the case of transforming another type of file format, the mapping files, described in RML format, belonging to the transformation of the specific format will be searched, and then the transformation to RDF will be performed using Helio[2].
6. After the conversion, if necessary between different file formats, a link between the different files will be made to have relations between the existing information in the different file formats transformed to RDF.
7. Once the linking has been done, or if not done, once the RDF conversion has been completed, a semantic and completeness check, is being performed on the generated RDF files via the RDF file validator of the Knowledge Graph generator to ensure consistency with the defined COGITO ontology.
8. Finally the RDF file is merged to the greater RDF graph file inside the triple-store which together with the OBJ file containing the geometric content of BIM extracted from step 4, form the COGITO data model. The Thing Description belonging to the project in which the new files and their transformations have been introduced will also be updated, validated, and stored in the thing directory.

To increase the data model query response time, the geometric content of the input BIM data is translated, via the B-rep generator component of the Data Post-Processing layer of DTP, to a graphics-friendly data following an open format such as OBJ (step 4 of the previous process). These geometric data in OBJ format together with the generated RDF graph constitute the

[2]https://github.com/helio-ecosystem

overall COGITO data model. The link between the geometric data contained in the OBJ file and the RDF graph data for every physical element of the BIM data is the IFC GUID. For non-BIM data, the process will be carried out without the BIM_Management component.

5. COGITO's digital twin query

The populated model can be queried. The queries can be classified into two categories:

- Simple queries, where the response is formed by looking at the semantic graph of the model. The list of elements completed during a specific time interval is an example of a simple query.
- Complex queries, where the response can be given by executing internal tools of DTP belonging to the data Post-processing layer.

Simple queries involve the Data Persistence, Data Management, and Messaging layers of DTP. The queries of the COGITO tools are formed as JSON messages which are translated into SPARQL query messages by the Data Management layer. These query messages are then transferred to the data persistence layer where the linked semantic graph is stored in the triple store. The response to these queries is then formed and transferred through the Data Management layer back to the tool which initiated the query as a response. A successful response message is then returned via the messaging layer to the respective tool as well. Complex queries are serviced similarly to simple queries involving also data post-processing layer execution calls.

6. Examples

The whole process of retrieving information from the RDF graph stored in the triplestore of the Data Persistence layer is illustrated in Figure 2. Initially, the RDF information and the related Project name are passed from the requesting non-DTP component to DTP's Data Management layer. Then, a first SPARQL query is formed in the Data Management layer of DTP (selecting the project by name) to the Thing Directory, to obtain back as the response the RDF URI of the file or information requested. Then, a second query (List.1) is formed in the Data Management layer of DTP and sent to the triple-store together with the RDF URI returned in the previous step. A final response to the second query is formed in JSON-LD 1.1 format and passed back to the requesting component via the Data Management Layer of DTP.

```
CONSTRUCT { ?s ?p ?o } WHERE {
    GRAPH <http://data.cogito.iot.linkeddata.es/resources/project/Project_1> {
            ?s ?p ?o
    } .
}
```

Listing 1: SPARQL query example to retrieve knowledge graph information requested

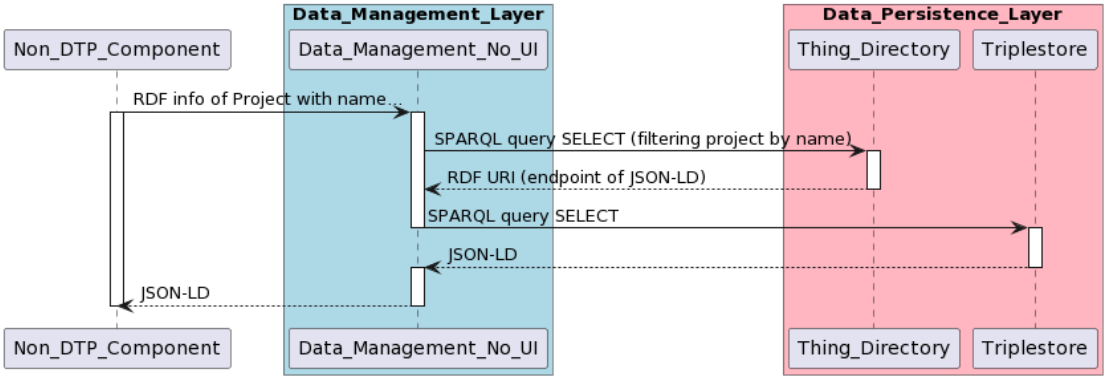

Figure 2: Sequence diagram to retrieve RDF information in JSON-LD

In addition, as we can see in Figure 3, a link between different types of files has been established in the formed RDF graph. This link is formed between information belonging to a Schedule file and a BIM file. In this figure, we can see that the formed JSON-LD (serialized from turtle) is presented in different colors. The blue part belongs to the BIM data (IFC), described in the graph in turtle format. The red part belongs to the Schedule data, also described in the graph in turtle format. As mentioned before, the Schedule data allows adding the fourth (time) dimension to the BIM data, achieving 4D-BIM data. This linking process is demonstrated in Figure 3 by, the green links, in the case of linking-by-element, and the purple links, in the case of linking-by-task. An example of linking between a task belonging to a Schedule file and an element belonging to the IFC file is also demonstrated in the same figure.

7. Conclusions

The generation and access of a semantically linked data model produced by a Digital Twin platform were introduced and analyzed. This model is the common data pool of an ICT framework called COGITO, which can be used to monitor a construction project in its design and implementation stages. The model fuses diverse data from different sources, in different open formats, such as IFC and OBJ, under a linked semantic graph in RDF format. This amalgamation provides the necessary abstraction capabilities for data access, interoperability, and transparency operations, required among the COGITO's internal tools and applications. These data model creation and access operations are demonstrated in an example referring to a building construction project, where data between the input IFC files (three-dimensional data) with a schedule and resource data (fourth dimension) are linked. Finally, our efforts to fuse such diverse data across the different fields of the construction sector revealed the need to extend existing ontological schemes, as well as to set boundaries to the type of data that can be converted to semantic graphs. Furthermore, apart from the IoT data, most of COGITO's input data, which are converted to RDF, are considered to be static. Changes in COGITO's input data, defined as differences in the form of deltas, are a topic of future investigation.

```
{
    "@context" : "schedule:construction",
    "@id" : "resources:construction/Construction001",
    "@type" : "Construction",
    "ifc_identifier" : "453",
    "has_elements" : [{
        "@id" : "resources:element:500",
        "@type" : "Element",
        "ifc_identifier" : "500",
        "involves_trask" : "resources:task/900"
    }],
    "has_spaces" : [{
        "@id" : "resources:space/342",
        "@type" : "Space",
        "ifc_identifier" : "342",
        "has_element" : "resources:element/500"
        },
        {
        "@type" : "Space",
        "ifc_identifier" : "345"
        }],
    "has_zone" : {
        "@type" : "Zone",
        "ifc_identifier" : "221",
        "has_space" : "resources:space/342",
        "has_element" : "resources:/element/500"
    },
    "is_related_to_process" : {
        "@type" : "Process",
        "process_id" : "1024",
        "has_task" : [{
            "@id" : "resources:task/900",
            "@type" : "Task",
            "task_identifier" : "900",
            "has_name" : "Task_900",
            "involves_resource" : "resources:resource/624",
            "involves_element" : "resources:element/500"
        }],
        "has_resources" : {
            "@id" : "resources:resource/624",
            "@type" : "Resource",
            "resource_identifier" : "624"
        }
        ...
    }
}
```


Figure 3: JSON-LD Example with relations between the information (IFC:Blue, Schedule:Red, Relations:Green and Purple)

8. Acknowledgements

The research leading to these results has been funded by the European Commission H2020-EU.2.1.5.2. project “COnstruction-phase diGItal Twin mOdel” under contract #958310 (COGITO).

References

[1] F. Tao, M. Zhang, A. Y. C. Nee, Digital twin driven smart manufacturing, Academic Press, 2019.

[2] A. Bolton, L. Butler, I. Dabson, M. Enzer, M. Evans, T. Fenemore, F. Harradence, E. Keaney, A. Kemp, A. Luck, et al., Gemini principles (2018).

[3] G. B. Ozturk, Digital twin research in the AECO-FM industry, Journal of Building Engineering 40 (2021) 102730.

[4] M. Shahzad, M. T. Shafiq, D. Douglas, M. Kassem, Digital twins in built environments: An investigation of the characteristics, applications, and challenges, Buildings 12 (2022) 120.

[5] R. Sacks, I. Brilakis, E. Pikas, H. S. Xie, M. Girolami, Construction with digital twin information systems, Data-Centric Engineering 1 (2020).

[6] S. Azhar, Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry, Leadership and management in engineering 11 (2011) 241–252.

[7] buildingSmart, ISO 16739-1:2018 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries — Part 1: Data schema, https://www.iso.org/standard/70303.html, 2018.

[8] S.-H. Lee, B.-G. Kim, IFC extension for road structures and digital modeling, Procedia Engineering 14 (2011) 1037–1042.

[9] M. Soilán, A. Nóvoa, A. Sánchez-Rodríguez, A. Justo, B. Riveiro, Fully automated methodology for the delineation of railway lanes and the generation of IFC alignment models using 3D point cloud data, Automation in Construction 126 (2021) 103684.

[10] N. Yabuki, E. Lebegue, J. Gual, T. Shitani, L. Zhantao, International collaboration for developing the bridge product model IFC-Bridge, in: 11th Int. Conf. on Computing in Civil and Building Engineering, 2006.

[11] M. Teodorovic, T. Hartmann, R. Tomar, I. Koulalis, J. Priedmore, K. Gavin, R. A. Chacón Flores, D1. 1 Launch Version of ASHVIN Platform (Version v0. 2), https://doi.org/10.5281/zenodo.4556836, 2021.

[12] R. Alonso, M. Borras, R. H. Koppelaar, A. Lodigiani, E. Loscos, E. Yöntem, SPHERE: BIM digital twin platform, in: Multidisciplinary Digital Publishing Institute Proceedings, volume 20, 2019, p. 9.